Inductive Proof Search Modulo

Fabrice Nahon, Claude Kirchner, Hélène Kirchner, Paul Brauner

Noname manuscript No.
(will be inserted by the editor)

Inductive Proof Search Modulo

Fabrice Nahon · Claude Kirchner · Hélène Kirchner · Paul Brauner

the date of receipt and acceptance should be inserted later

Contents

1 Introduction 2
2 A motivating example 4
3 Equational rewriting and narrowing : basic concepts and results 5
4 Deduction modulo and inductive proofs 8
5 A proof search system for induction modulo 12
6 Induction modulo AC and A 17
7 Prototype and computer experiments 25
8 Conclusion 30

Abstract We present an original narrowing-based proof search method for inductive theorems in equational rewrite theories given by a rewrite system $\mathcal{R}$ and a set E of equalities. It has the specificity to be grounded on deduction modulo and to rely on narrowing to provide both induction variables and instantiation schemas. Whenever the equational rewrite system $(\mathcal{R}, E)$ has good properties of termination, sufficient completeness, and when E is constructor and variable preserving, narrowing at defined-innermost positions leads to consider only unifiers which are constructor substitutions. This is especially interesting for associative and associative-commutative theories for which the general proof search system is refined. The method is shown to be sound and refutationaly correct and complete. A major feature of our approach is to provide a constructive proof in deduction modulo for each successful instance of the proof search procedure.

Keywords Deduction modulo · Noetherian induction · equational rewriting · equational narrowing.

Fabrice Nahon
LORIA and Rectorat Nancy-Metz, France E-mail: Fabrice.Nahon@loria.fr

Claude Kirchner
INRIA, France E-mail: Claude.Kirchner@inria.fr

Hélène Kirchner
INRIA, France E-mail: Helene.Kirchner@inria.fr

Paul Brauner
LORIA and University of Nancy, France E-mail: Paul.Brauner@loria.fr

1 Introduction

Proof by induction is a main mathematical reasoning principle and is of prime interest in informatics. Typically in hardware and software verification problems, when dealing with security protocols or safety properties of embedded systems, reasoning on complex data structures, with an infinite number of objects or states, makes a prominent use of induction.

Three main approaches have been developed for mechanizing inductive proofs. The first one, *explicit induction*, is used in proof assistants, for instance Nqthm-ACL2 [KAU 96], Coq [BER 04], Isabelle[NIP 02] or Inka [AUT 99]. Explicit induction uses structural induction on data types and *induction rules* as for example Peano induction. These forms of induction are in fact subsumed by the single general schema of Noetherian induction, called *noetherian induction principle*. It is based on the noetherianity of a relation $<$ on a set τ stating that there is no infinite decreasing sequence of elements in τ. The *Noetherian induction principle* states that if, for any element $x \in \tau$, a proposition P holds for x whenever it holds for all elements $\underline{x}$ such that $\underline{x} < x$, then P holds for all $x \in \tau$. Mechanizing proof by induction [BUN 99] is hard due to the intrinsic difficulty of finding the most convenient inductive rule to show a given conjecture. Indeed there is an infinite variety of possible noetherian relations and choosing an appropriate induction rule introduces a first branching point in the search space. Furthermore, such proofs involve in general two tasks: generalizing the induction formula and introducing an intermediate lemma. In Gentzen's original formalization of sequent calculus, this is an instance of the cut rule that can be applied to any formula, thus introducing a second infinite branching point in the search space. Unfortunately, Kreisel has shown that Gentzen's cut rule is not redundant for inductive theories. In fact, cut-elimination is possible in the presence of induction, provided the induction rule is formulated so as to allow arbitrary induction hypotheses to be introduced (see e.g McDowel and Miller for the intuitionist case and Brotherston and Simpson for the classical case [BRO 07]).

The second approach, *induction by consistency* (or *inductionless induction*), roughly works as follows: given a set of clauses $\mathcal{E}$ and a set of conjectures $\mathcal{C}$, we add $\mathcal{C}$ to $\mathcal{E}$ and run a deduction engine until one gets a saturated and consistent set of clauses. Historically, this deduction engine was given by the Knuth-Bendix completion procedure [KNU 70] (see [MUS 80] and [GOG 80]) and worked when $\mathcal{E}$ and $\mathcal{C}$ were both sets of equalities. The procedure attempts to complete the initial system $\mathcal{C} \cup \mathcal{E}$ by iteratively adding new equalities called critical pairs: these critical pairs are obtained by superposing an equality of the corresponding system into another one, in all possible ways. One main problem is that such a completion often loops, generating infinitely many critical pairs. Fribourg [FRI 86] first observed that only overlaps of axioms on conjectures are necessary: it is the so-called "linear strategy". Moreover, whenever $\mathcal{E}$ can be turned into a ground convergent and sufficiently complete rewrite system, overlaping can be performed at specific positions. Comon and Nieuwenhuis [COM 00] gave a more general view of the deduction system, without restricting the conjectures to equalities. The interested reader may refer to [COM 01, section 1.3] for all relevant references on this approach.

The last approach, *implicit induction* (or *induction by rewriting*), is used in automated theorem provers like Spike [BOU 92] or RRL [KAP 95]. The main idea of implicit induction is as follows: given a terminating rewrite system, the corresponding rewrite relation is noetherian and can be used for induction. Reddy [RED 90] provided

a method to prove equalities in this way. The pragmatic advantage of his method is that it does not need to explicitly check that the inductive hypothesis is applied to smaller terms. Kounalis and Rusinowitch [BOU 95] designed a proof technique which works in conditional rewrite systems and applies to non-Horn clauses as well: it is the so-called *test set induction*. It works by computing an appropriate implicit induction scheme called a test set and then applies a refutation principle using proof by consistency techniques. Their method is refutationaly complete and can refute false conjectures even in the cases where the functions are not completely defined. It has been implemented in the prover Spike.

In this large field of proof by induction, our contribution is three-fold. First, as a bridge between explicit and implicit induction, we provide a proof search mechanism for inductive proofs, previously explored in [DEP 03, DEP 02, KIR 06], relying on the deduction modulo approach [DOW 03]. This latter semi-decision procedure comes along with a constructive proof of soundness which associates a proof in deduction modulo to every successful instance of the algorithm presented here. This major feature opens the door to formal checking of the algorithm's results and to its integration as a tactic in proof assistants that require some proof witness like Coq.

Then, although already quite expressive, this approach was first designed for theories expressed by rewrite rules and is thus limited by the fact that axioms like commutativity cannot be oriented as a rule without losing termination of the underlying rewrite system. The solution consists then of using equational rewriting (also called rewriting modulo) as pioneered by [PET 81] and [JOU 86] and to extend the proof search method developed in [DEP 03] in order to perform induction in theories containing such non orientable axioms. This extension should also be compared to implicit induction techniques used for induction modulo associativity and commutativity as done in [BER 96] and [AOT 06] who generalizes [RED 90]. The second main contribution of this paper is to show that under some conditions, it is sufficient to perform unification at *defined innermost* positions, i.e. positions the subterms under which are only constituted of variables and constructor symbols. Hence, serious difficulties, related to the size of complete sets of unifiers, can be avoided. For instance, it becomes possible to perform induction modulo non finitary theories. In particular, unification modulo associativity and commutativity boils down to unification modulo commutativity while unification modulo associativity is reduced to syntactic unification.

Finally the third important contribution to emphasize is that the procedure is proved refutationaly correct and complete. In particular this semi-procedure terminates with failure when launched against some false assertion.

The remainder of this paper is built as follows: the next section provides on a simple arithmetic example an informal introduction of technical concepts and of the general method. Section 3 recalls basic results about rewriting and narrowing and introduces the concepts of *constructor preserving* theories, *defined-innermost* positions and *complete sets of constructor unifiers* that are used in the following. Section 4 presents the logical framework of deduction modulo which is necessary to understand the proof construction and the proof search mechanisms. In particular, we explain how deduction modulo manages the Noetherian induction principle. In Section 5, we present the proof search system for inductive proofs modulo a general theory E, which is proved sound and refutationaly correct and complete. Section 6 deals with the special case of associative-commutative theories or associative theories. The proof system of Section 5 is instantiated in these cases with more operational proof steps. The method is implemented in a prototype prover described in Section 7 as well as several examples.

2 A motivating example

In the following we sketch how our method works on a simple example and show how it is essentially different from previous ones.

- **Sorts:** nat;
- **constructors:** $0 : \rightarrow nat \qquad s : nat \rightarrow nat$
- **defined functions:** $+ : nat \times nat \rightarrow nat \qquad * : nat \times nat \rightarrow nat$
- **rules**:

$$\begin{array}{llllll} x+0 & \rightarrow x & x*0 & \rightarrow 0 & exp(x,0) & \rightarrow s(0) \\ x+s(y) & \rightarrow s(x+y) & x*s(y) & \rightarrow x*y+x & exp(x,s(y)) & \rightarrow x*exp(x,y) \end{array}$$

Fig. 1 Simple arithmetic

In the theory *Simple arithmetic* described in Figure 1, let us try to prove the proposition

$$\forall x, y, n \; exp(x*y, n) \approx exp(x, n) * exp(y, n)$$

where $+$ and $*$ are also assumed to be associative and commutative (AC). The method developed in [BER 97] is based on *induction schemes*. More precisely, it computes a subset of variables of the goal, the *induction variables*, and a set of terms, the *test set*. The induction variables are replaced by elements of the test set, and such replacements produce new conjectures which are simplified by rewrite rules of the specification and smaller instances of the original conjecture (the induction hypothesis). The proof is completed when all newly generated conjectures are simplified into known or trivial inductive theorems. Algorithms are provided to compute induction variables and test sets. Notice that Boyer and Moore [BOY 79] were the first to introduce the notion of "machine" that computes an appropriate subset of variables as induction variables. In our example, the induction variables are x, y, and n, and the test set is $\{0, s(x)\}$. Therefore, a *test instance* is $exp(s(x') * s(y'), s(n')) \approx exp(s(x'), s(n')) * exp(s(y'), s(n'))$. However, this last equality can be reduced by rules of the specification into $s(x') * s(y') * exp(s(x'+y'+x'*y'), n') \approx s(x') * exp(s(x'), n') * s(y') * exp(s(y'), n')$, which cannot be simplified by the induction hypothesis, and the proof attempt may fail. One can avoid this difficulty if the set of induction variables is restricted. That is why in [BER 97], the authors have defined an heuristic in order to select *good* induction variables relying on observations of the Nqthm-ACL2 system. Using this strategy on the example, only the variable n is instantiated and the proof search succeeds. However, the method does not remain refutationaly complete with such an heuristic.

In our approach, the induction step is performed by narrowing at *defined-innermost* positions, when the theory is axiomatized by a sufficiently complete and terminating equational rewrite system. More precisely, it suffices to perform the narrowing step at only one defined-innermost position. In the above situation, the defined innermost positions are 1.1, 2.1 and 2.2, which respectively denote the subterms $exp(x * y, z)$, $exp(x, n)$ and $exp(y, n)$. Now, since $*$ is commutative, the goal remains equivalent by permuting the variables x and y, therefore two possibilities remain: narrowing at the defined-innermost position 1.1 where the symbol $*$ occurs, or 2.1 where the symbol exp occurs. We consider the latter better, since it further creates more reductions,

we choose to narrow at the position 2.1. After one narrowing step, the subgoals are $exp(x*y,0) \approx s(0)*exp(y,0)$ and $exp(x*y,s(n)) \approx x*exp(x,n)*exp(y,s(n))$. After normalization, we obtain the trivial subgoal $s(0) \approx s(0)$ and $x*y*exp(x*y,n) \approx x*y*exp(x,n)*exp(y,n)$ which can be reduced by the induction hypothesis. This therefore finishes the proof.

3 Equational rewriting and narrowing : basic concepts and results

For the main notations and classical results on term rewriting, we refer for instance to [BAA 98] or [KIR 99]. We recall here shortly our main notations.

In a many sorted signature $(\mathcal{S}, \Sigma)$ (or simply Σ, for short) where $\mathcal{S}$ is a set of *sorts* and Σ is a set of function symbols, each symbol f is given with a rank $f : S_1 \times \ldots \times S_n \rightarrow S$, where $S_1, \ldots, S_n, S \in \mathcal{S}$ and n is the arity of f. We assume moreover that the signature Σ comes in two parts, $\Sigma = \mathcal{C} \cup \mathcal{D}$, where $\mathcal{C}$ is a set of *constructor symbols*, and $\mathcal{D}$ is a set of *defined symbols*. A constructor term is a term built only with constructor symbols. Let $\mathcal{X}$ be a family of sorted variables. The set of well-sorted terms over Σ (resp. well-sorted constructor terms) with variables in $\mathcal{X}$ is denoted by $\mathcal{T}(\Sigma, \mathcal{X})$ (resp. $\mathcal{T}(\mathcal{C}, \mathcal{X})$). The subset of $\mathcal{T}(\Sigma, \mathcal{X})$ (resp. $\mathcal{T}(\mathcal{C}, \mathcal{X})$) of variable-free terms, or *ground* terms, is denoted $\mathcal{T}(\Sigma)$ (resp. $\mathcal{T}(\mathcal{C})$). A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is identified as usual to a function from its set of *positions* (strings of positive integers) $\mathcal{D}om(t)$ to symbols of Σ and $\mathcal{X}$. We note ε the empty string (root position). The *subterm* of t at position ω is denoted by $t_{|\omega}$, and $t(\omega)$ denotes its head symbol. The result of replacing $t_{|\omega}$ with s at position ω in t is denoted by $t[s]_\omega$. This notation is also used to indicate that s is a subterm of t and, in this case, the position ω may be omitted. $\mathcal{V}ar(t)$ denotes the set of (free) variables of the term t and $|\mathcal{V}ar(t)|$ its cardinality. We define $\overrightarrow{\mathcal{V}ar(t)}$ as the vector of variables assumed linearly ordered. These notations are extended to formulas of the form $t_1 \# t_2$ seen as terms with top symbol $\#$ of arity 2 and to sets of such formulas. A substitution is a finite mapping $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ where $x_1, \ldots, x_n \in \mathcal{X}$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$. We use postfix notation for substitutions application and composition. The domain of a substitution σ is the set $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid x\sigma \neq x\}$, the set of variables introduced by σ is the set $\mathcal{R}an(\sigma) = \bigcup_{x \in \mathcal{D}om(\sigma)} \mathcal{V}ar(x\sigma)$, and the image of σ is the set $Im(\sigma) = \{t \in \mathcal{T}(\Sigma, \mathcal{X}) \mid \exists x \in \mathcal{D}om(\sigma),\ t = x\sigma\}$. A substitution σ is ground whenever $Im(\sigma) \subseteq \mathcal{T}(\Sigma)$, and is constructor whenever $Im(\sigma) \subseteq \mathcal{T}(\mathcal{C}, \mathcal{X})$. Given two terms s and t, a *unifier* of s and t is a substitution σ such that $s\sigma = t\sigma$, and a *most general unifier* of s and t ($mgu(s,t)$ for short) is a unifier σ such that, for any unifier θ of s and t, there exists a substitution μ such that $\theta = \sigma\mu$ on the variables of s and t.

Given a relation $\rightarrow$ on $\mathcal{T}(\Sigma, \mathcal{X})$, $\stackrel{+}{\rightarrow}$ and $\stackrel{*}{\rightarrow}$ denote the transitive and the reflexive transitive closure of $\rightarrow$ respectively. A normal form of t, denoted $t\downarrow$, is such that $t \stackrel{*}{\rightarrow} t\downarrow$ and $t\downarrow$ cannot be reduced by the relation $\rightarrow$. The normalized form $\sigma\downarrow$ of a substitution σ is defined by $x(\sigma\downarrow) = (x\sigma)\downarrow$ for all $x \in \mathcal{D}om(\sigma)$. An equality is an expression of the form $t_1 \approx t_2$, where t_1 and t_2 are two terms of the same sort. Given a set E of equalities, $=_E$ denotes the congruence generated by E. Equalities are symmetric, i.e. we make no difference between $t_1 \approx t_2$ and $t_2 \approx t_1$.

Given two terms s and t, an *E-unifier* of s and t is a substitution σ such that $s\sigma =_E t\sigma$, and a *complete set* of E-unifiers of s and t ($CSU_E(s,t)$ for short) is a set of E-unifiers of s and t satisfying: for any E-unifier θ of s and t, there exist $\sigma \in CSU_E(s,t)$

and a substitution μ such that $\theta =_E \sigma\mu[\mathcal{V}ar(s) \cup \mathcal{V}ar(t)]$, i.e. $\theta(x) =_E \sigma\mu(x)$ for all $x \in \mathcal{V}ar(s) \cup \mathcal{V}ar(t)$.

Definition 1 A set E of equalities is *regular* if for any equality $t_1 \approx t_2 \in E$, $\mathcal{V}ar(t_1) = \mathcal{V}ar(t_2)$. A set E of equalities is *constructor preserving* whenever E is regular, and, for any equality $t_1 \approx t_2 \in E$, $t_1 \in \mathcal{T}(\mathcal{C}, \mathcal{X}) \Rightarrow t_2 \in \mathcal{T}(\mathcal{C}, \mathcal{X})$.

As a consequence of this definition, a set E of equalities is constructor preserving if and only if two terms cannot be E-equivalent whenever one of them is constructor and the other is not. Typically, if $+ \in \mathcal{D}$ and $0 \in \mathcal{C}$, $E = \{0 + x = x\}$ is not constructor preserving (since $0 + 0 = 0$), but associativity or commutativity of $+$ are.

3.1 Equational rewriting and narrowing

Let us turn now to rewriting modulo and we refer to [JOU 86] for a full exposition. A rewrite rule is an ordered pair of terms $l \to r$ such that $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ and l is not a variable. A conditional rewrite rule $c \Rightarrow l \to r$ moreover satisfies $\mathcal{V}ar(c) \subseteq \mathcal{V}ar(l)$. A rewrite system $\mathcal{R}$ is a set of rewrite rules. An *equational rewrite system* is given by a set of rewrite rules $\mathcal{R}$ and a set of equalities E. Let $\to_{\mathcal{R}/E}$ ($\mathcal{R}/E$ for short) be the relation $=_E \circ \to_{\mathcal{R}} \circ =_E$ which simulates the relation induced by $\mathcal{R}$ in E-equivalence classes.

Definition 2 An equational rewrite system $(\mathcal{R}, E)$ is *terminating modulo* E if the relation $\mathcal{R}/E$ is Noetherian, i.e. all rewrite derivations are finite. It is *ground terminating modulo* E if it is terminating modulo E over the set of ground terms.

Given an equational rewrite system $(\mathcal{R}, E)$, the rewriting modulo E relation $\to_{\mathcal{R},E}$ ($\mathcal{R}, E$ for short) and the narrowing modulo E relation $\leadsto_{\mathcal{R},E}$ are defined as follows:

Definition 3 Given two terms s, $t \in \mathcal{T}(\Sigma, \mathcal{X})$, s *rewrites modulo* E *to* t, denoted $s \to_{\mathcal{R},E} t$, whenever there exist a rewrite rule $l \to r \in \mathcal{R}$, a position $\omega \in \mathcal{D}om(t)$, and a substitution σ, such that $s_{|\omega} =_E l\sigma$ and $t = s[r\sigma]_\omega$. In this case, s is said $\mathcal{R}, E$-reducible. In addition, for a conditional rule $c \Rightarrow l \to r$, $c\sigma$ must evaluate to true when applying the rule. Also, s *narrows modulo* E *into* t, denoted $s \leadsto_{\mathcal{R},E} t$, whenever there exist a rewrite rule $l \to r \in \mathcal{R}$, a position $\omega \in \mathcal{D}om(t)$, and a substitution σ, such that $s_{|\omega}\sigma =_E l\sigma$ and $t = (s[r]_\omega)\sigma$.

Since $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R},E} \subseteq \to_{\mathcal{R}/E}$, termination of $\mathcal{R}/E$ implies termination of $\to_{\mathcal{R}}$ and $\to_{\mathcal{R},E}$. Sufficient completeness is a fundamental property which states that it is always possible to rewrite any ground non-constructor term into a constructor one:

Definition 4 A relation $\to$ is *sufficiently complete modulo* E when, for any $s \in \mathcal{T}(\Sigma)$, there exists $t \in \mathcal{T}(\mathcal{C})$, such that $s \stackrel{*}{\to} t$. The equational rewrite system $(\mathcal{R}, E)$ is *sufficiently complete modulo* E if the relation $\to_{\mathcal{R},E}$ is.

For ground terminating and sufficiently complete modulo E rewrite systems, it is possible to specify particular positions in terms where reductions must apply, and where case analysis by rewriting can usefully be done.

Definition 5 For any $t \in \mathcal{T}(\Sigma, \mathcal{X})$, a position ω in t is called *defined-innermost*, and we write $\omega \in DI(t)$, if $t(\omega) \in \mathcal{D}$ and $t(\omega') \in \mathcal{C} \cup \mathcal{X}$ whenever $\omega < \omega'$, where $<$ denotes the prefix ordering.

For instance, considering the Peano's integers defined in the simple arithmetic example of Fig. 1, in $s((0+0)+s(0+s(x)))$, the positions 1.1 and 1.2.1 are defined-innermost but 1 is not.

The following proposition states that defined-innermost positions are ground $\mathcal{R}, E$-reducible under appropriate assumptions.

Proposition 1 Assume that $(\mathcal{R}, E)$ is sufficiently complete modulo E and that E is constructor preserving. Then, for any term t, for any ground $\mathcal{R}, E$-normalized substitution α, and for any $\omega \in DI(t)$, $t\alpha$ is $\mathcal{R}, E$-reducible at position ω.

Proof. Let $f = t(\omega)$. There exist constructor terms $t_1, \ldots, t_n$ such that $t_{|\omega} = f(t_1, \ldots, t_n)$. Let α be a ground $\mathcal{R}, E$-irreducible instance of $t_{|\omega}$. Since $f \in \mathcal{D}$, $(t_{|\omega})\alpha$ is not a constructor term, and since $\mathcal{R}, E$ is sufficiently complete, $(t_{|\omega})\alpha$ is not a $\mathcal{R}, E$-normal form. Therefore, there exist a position $\omega' \in \mathcal{D}om((t_{|\omega})\alpha)$, a rewrite rule $l \to r \in \mathcal{R}$, and a substitution ν, such that $((t_{|\omega})\alpha)_{|\omega'} =_E l\nu$. Now, let us distinguish two cases.

1. $\omega' \neq \varepsilon$. Then $((t_{|\omega})\alpha)_{|\omega'} \in \mathcal{T}(\mathcal{C})$, since α is ground $\mathcal{R}, E$-normalized, $\omega \in DI(t)$, and $\mathcal{R}, E$ is sufficiently complete. However, this leads to a contradiction, since E is constructor preserving.
2. $\omega' = \varepsilon$. Then $(t_{|\omega})\alpha =_E l\nu$, thus $(t_{|\omega})\alpha$ is $\mathcal{R}, E$-reducible and we are done.

□

3.2 Constructor E-unifiers

A main difference between previous narrowing or superposition-based approaches and the one proposed in this paper, is that the unification used here to perform narrowing is quite restricted. For instance, when reasoning modulo associativity, instead of considering potentially infinite sets of unifiers, we can safely restrict to finitely many ones.

For a given set E of equalities, *constructor E-unifiers* are a key to tame the proof search system IndNarrowModE presented below. *Complete sets of constructor E-unifiers* are generating sets of constructor unifiers:

Definition 6 Given two terms $s,\ t \in \mathcal{T}(\Sigma, \mathcal{X})$, a substitution σ is a constructor E-unifier of s and t if $s\sigma =_E t\sigma$ and $Im(\sigma) \subseteq \mathcal{T}(\mathcal{C}, \mathcal{X})$. The set of E-unifiers $CSUC_E(s,t)$ is a *complete set of constructor E-unifiers* of s and t, if:

Correctness: every σ of $CSUC_E(s,t)$ is a constructor E-unifier of s and t;
Completeness: for any constructor E-unifier θ of s and t, there exists $\sigma \in CSUC_E(s,t)$ and a substitution μ, such that $\theta =_E \sigma\mu\ [\mathcal{V}ar(s) \cup \mathcal{V}ar(t)]$;
Domain: for any $\sigma \in CSUC_E(s,t)$, $\mathcal{R}an(\sigma) \cap \mathcal{D}om(\sigma) = \emptyset$.

If E is constructor preserving and satisfies syntactic conditions detailed in [NAH 07], the subset of all constructor elements of $CSU_E(s,t)$ is a complete set of constructor E-unifiers of s and t. This is in particular the case when considering some associative (A) or associative and commutative (AC) theories. More precisely, when E is an AC theory involving only defined symbols, if s and t are terms and ω is a defined-innermost position in s, then $CSUC_E(s_{|\omega}, t)$ is $CSUC_C(s_{|\omega}, t)$, where C denotes the subset of commutativity axioms of E. In other words, in this case AC

constructor unification reduces to C constructor unification. Similarly if E is an associative theory involving only defined symbols, although $CSU_E(s_{|\omega}, t)$ may be infinite, $CSUC_E(s_{|\omega}, t)$ is $CSUC_\emptyset(s_{|\omega}, t)$ which of course considerably reduces the complexity of the unification problem.

To conclude this section, the following proposition shows that, whenever E is constructor preserving and $(\mathcal{R}, E)$ is sufficiently complete modulo E, the narrowing step at defined-innermost positions is performed with constructor substitutions:

Proposition 2 Assume that $(\mathcal{R}, E)$ is sufficiently complete modulo E and that E is constructor preserving. Then, for all $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, for any $f \in \mathcal{D}$, for any ground $\mathcal{R}, E$-normalized instantiation α of $f(t_1, \ldots, t_n)$, and for any set V such that $\mathcal{D}om(\alpha) \subseteq V$, there exist a rewrite rule $l \to r \in \mathcal{R}$, a substitution $\sigma \in CSUC_E(f(t_1, \ldots, t_n), l)$ and a substitution μ such that: $\sigma\mu =_E \alpha \quad [V]$.

Proof. Since α is ground $\mathcal{R}, E$-normalized, $\mathcal{R}, E$ is sufficiently complete modulo E, $f \in \mathcal{D}$, and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, then, by Proposition 1, $f(t_1, \ldots, t_n)\alpha$ is $\mathcal{R}, E$-reducible at position ε. By the classical lifting lemma [HUL 80, KIR 99], there exist a rewrite rule $l \to r \in \mathcal{R}$, a $E-$unifier σ of $f(t_1, \ldots, t_n)$ and l, and a substitution μ, such that $\sigma\mu =_E \alpha[V]$. Now, it remains to show that σ is a constructor substitution. Let $x \in V$. We have $x\sigma\mu =_E x\alpha$, and, since α is ground $\mathcal{R}, E$-irreducible and $\mathcal{R}, E$ is sufficiently complete, this leads to $x\alpha \in \mathcal{T}(\mathcal{C})$. The conclusion follows, since E is assumed to be constructor preserving. □

Thanks to these settings, we present in what follows an inductive proof search system, relying on a main induction rule that uses narrowing to choose both the induction variables and the instantiation schema.

4 Deduction modulo and inductive proofs

Let us now explain in this section how deduction modulo can provide the description, at the proof theoretical level, of proof by Noetherian induction.

Because we quantify on propositions, the Noetherian induction principle is by essence a second-order proposition. Since we want to make a primarily use of first-order rewrite concepts and techniques and to consider first-order theories, we need a first-order presentation of higher-order logic. We use the so-called $HOL_{\lambda\sigma}$ introduced in [DOW 01] which is based on deduction modulo [DOW 03] and reveals to be particularly well-suited for our concerns. It is clearly out of the scope of this paper to explain in detail the full approach, and we only sketch here the main ideas. The reader can refer to [DEP 02] and to [DEP 04] for a detailed exposition.

4.1 Deduction modulo

In deduction modulo, terms but also propositions can be identified modulo a congruence. We use a congruence that can typically be defined by conditional equalities and that takes into account the application context to evaluate the conditions. Let us consider theories described by a set of axioms Γ and a congruence, denoted $\sim$, defined on terms and propositions. The deduction rules of the sequent calculus take this equivalence into account. For instance, the right rule for the conjunction is not stated as usual

$$\frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}$$

but is formulated

$$\frac{\Gamma \vdash_{\sim} A, \Delta \quad \Gamma \vdash_{\sim} B, \Delta}{\Gamma \vdash_{\sim} D, \Delta} \text{ if } D \sim A \wedge B.$$

We recall in Figure 2, the definition of the *sequent calculus modulo*. In these rules, Γ and Δ are finite multisets of propositions, P and Q denote propositions. Substituting the variable x by the term u in Q is denoted $Q\{u/x\}$. When the congruence $\sim$ is simply identity, this sequent calculus collapses to the usual one [GIR 89]. In that case, sequents are written as usual with the $\vdash$ symbol.

4.2 Deduction modulo for inductive proofs

This short introduction to deduction modulo now allows us to give a proof theoretic understanding of induction by rewriting. In the context of deduction modulo, the induction hypotheses arising from equational goals can be (dynamically) internalized into the congruence. When doing this, the computational part of the deduction modulo appears to perform induction by rewriting as done for instance by systems like Spike [BOU 92] or RRL [KAP 95].

A relation $\prec$ on a set τ is Noetherian (or well-founded) if all chains are of finite length. The *Noetherian induction principle* states that if, for any element $x \in \tau$, a proposition P holds for x whenever it holds for all $\underline{x}$ such that $\underline{x} \prec x$, then P holds for all $x \in \tau$. Let $NoethInd(P, \prec, \tau)$ be the following proposition:

$$\forall x\ (x \in \tau \wedge ((\forall \underline{x}\ \underline{x} \in \tau \wedge \underline{x} \prec x \Rightarrow P(\underline{x})) \Rightarrow P(x))) \Rightarrow \forall x\ (x \in \tau \Rightarrow P(x))$$

and the subformula $\forall \underline{x}\ \underline{x} \in \tau \wedge \underline{x} \prec x \Rightarrow P(\underline{x})$ is the induction hypothesis. Well-foundedness and the principle of noetherian induction are equivalent notions [WEC 92]. The pragmatic advantage is that the principle of noetherian induction based on the relation $\prec$, holds for all propositions P, whenever $\prec$ is noetherian. Formally, it is the proposition NI below:

$$NI:\ \forall \prec\ \forall \tau\, Noeth(\prec, \tau) \Rightarrow \forall P\ NoethInd(P, \prec, \tau).$$

where $Noeth(\prec, \tau)$ states that $\prec$ is a Noetherian relation over τ.

Proving that P inductively holds in a user theory Th_u, denoted $Th_u \models_{Ind} P$, amounts to derive the sequent $NI, Th_u \vdash P$ and to provide a proof of $Noeth(\prec, \tau)$. The whole problem is formalized in $HOL_{\lambda\sigma}$. The remainder of this section gives the main steps which are detailed in [DEP 02].

To fix the ideas, let us consider that the property to prove is of the form $\forall x\,(x \in \tau \Rightarrow Q(x))$ where $Q(x)$ is an equation $t_1(x) \approx t_2(x))$. We start from the sequent:

$$\begin{array}{l} \forall \prec\ \forall \tau\,(Noeth(\prec, \tau) \Rightarrow \forall P\, NoethInd(P, \prec, \tau)), Th_u \\ \vdash \\ \forall x\,(x \in \tau \Rightarrow Q(x)) \end{array}$$

Choosing a specific relation $\prec$ (written $<$) and a set denoted $\mathcal{T}$, we get:

$$Noeth(<, \mathcal{T}) \Rightarrow \forall P\, NoethInd(P, <, \mathcal{T}), Th_u \vdash \forall x\,(x \in \mathcal{T} \Rightarrow Q(x)).$$

$$\frac{}{\Gamma, P \vdash_\sim Q}\text{axiom}\ \ \text{if}\ P \sim Q \qquad \frac{\Gamma, P \vdash_\sim \Delta \quad \Gamma \vdash_\sim Q, \Delta}{\Gamma \vdash_\sim \Delta}\text{cut}\ \ \text{if}\ P \sim Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash_\sim \Delta}{\Gamma, P \vdash_\sim \Delta}\text{contr-l}\ \ \text{if}\ (\mathbf{A}) \qquad \frac{\Gamma \vdash_\sim Q_1, Q_2, \Delta}{\Gamma \vdash_\sim P, \Delta}\text{contr-r}\ \ \text{if}\ (\mathbf{A})$$

$$\frac{\Gamma \vdash_\sim \Delta}{\Gamma, P \vdash_\sim \Delta}\text{weak-l} \qquad \frac{\Gamma \vdash_\sim \Delta}{\Gamma \vdash_\sim P, \Delta}\text{weak-r}$$

$$\frac{\Gamma, P, Q \vdash_\sim \Delta}{\Gamma, R \vdash_\sim \Delta}\wedge\text{-l}\ \ \text{if}\ R \sim (P \wedge Q) \qquad \frac{\Gamma \vdash_\sim P, \Delta \quad \Gamma \vdash_\sim Q, \Delta}{\Gamma \vdash_\sim R, \Delta}\wedge\text{-r}\ \ \text{if}\ R \sim (P \wedge Q)$$

$$\frac{\Gamma, P \vdash_\sim \Delta \quad \Gamma, Q \vdash_\sim \Delta}{\Gamma, R \vdash_\sim \Delta}\vee\text{-l}\ \ \text{if}\ (\mathbf{B}) \qquad \frac{\Gamma \vdash_\sim P, Q, \Delta}{\Gamma \vdash_\sim R, \Delta}\vee\text{-r}\ \ \text{if}\ (\mathbf{B})$$

$$\frac{\Gamma \vdash_\sim P, \Delta \quad \Gamma, Q \vdash_\sim \Delta}{\Gamma, R \vdash_\sim \Delta}\Rightarrow\text{-l}\ \ \text{if}\ (\mathbf{C}) \qquad \frac{\Gamma, P \vdash_\sim Q, \Delta}{\Gamma \vdash_\sim R, \Delta}\Rightarrow\text{-r}\ \ \text{if}\ (\mathbf{C})$$

$$\frac{\Gamma \vdash_\sim P, \Delta}{\Gamma, R \vdash_\sim \Delta}\neg\text{-l}\ \ \text{if}\ R \sim \neg P \qquad \frac{\Gamma, P \vdash_\sim \Delta}{\Gamma \vdash_\sim R, \Delta}\neg\text{-r}\ \ \text{if}\ R \sim \neg P$$

$$\frac{}{\Gamma, P \vdash_\sim \Delta}\bot\text{-l}\ \ \text{if}\ P \sim \bot$$

$$\frac{\Gamma, Q\{t/x\} \vdash_\sim \Delta}{\Gamma, P \vdash_\sim \Delta}(Q, x, t)\ \forall\text{-l}\ \ \text{if}\ (\mathbf{D}) \qquad \frac{\Gamma \vdash_\sim Q\{y/x\}, \Delta}{\Gamma \vdash_\sim P, \Delta}(Q, x, y)\ \forall\text{-r}\ \ \text{if}\ (\mathbf{E})$$

$$\frac{\Gamma, Q\{y/x\} \vdash_\sim \Delta}{\Gamma, P \vdash_\sim \Delta}(Q, x, y)\ \exists\text{-l}\ \ \text{if}\ (\mathbf{F}) \qquad \frac{\Gamma \vdash_\sim Q\{t/x\}, \Delta}{\Gamma \vdash_\sim P, \Delta}(Q, x, t)\ \exists\text{-r}\ \ \text{if}\ (\mathbf{G})$$

$\mathbf{A} = P \sim Q_1 \sim Q_2$, $\mathbf{B} = R \sim (P \vee Q)$, $\mathbf{C} = R \sim (P \Rightarrow Q)$, $\mathbf{D} = P \sim \forall x\ Q$, $\mathbf{E} = P \sim \forall x\ Q, y$ fresh variable, $\mathbf{F} = P \sim \exists x\ Q, y$ fresh variable, $\mathbf{G} = P \sim \exists x\ Q$.

Fig. 2 The sequent calculus modulo

From this, by the rule $\Rightarrow$-l of the sequent calculus, we get on one hand the sequent $Th_u \vdash Noeth(<, \mathcal{T})$ corresponding to the proof that $<$ is indeed Noetherian on $\mathcal{T}$, on the other hand the sequent

$$\forall P\, NoethInd(P, <, \mathcal{T}), Th_u \vdash \forall x\, (x \in \mathcal{T} \Rightarrow Q(x))$$

corresponding to the use of the induction principle to prove our property.

We instantiate P as the goal to prove and we get:

$$\forall x\,((x \in \mathcal{T} \wedge \forall \underline{x}\,((\underline{x} \in \mathcal{T} \wedge \underline{x} < x) \Rightarrow Q(\underline{x}))) \Rightarrow Q(x))$$
$$\Rightarrow \forall x\,(x \in \mathcal{T} \Rightarrow Q(x)), Th_u \vdash \forall x\,(x \in \mathcal{T} \Rightarrow Q(x))$$

A few easy steps of the sequent calculus later, we get:

$$Th_u \vdash \forall x\,((x \in \mathcal{T} \wedge \forall \underline{x}\,((\underline{x} \in \mathcal{T} \wedge \underline{x} < x) \Rightarrow Q(\underline{x}))) \Rightarrow Q(x)).$$

We then instantiate x by a fresh variable called X to emphasize this status, and we get:

$$Th_u \vdash (X \in \mathcal{T} \wedge \forall \underline{x}\,((\underline{x} \in \mathcal{T} \wedge \underline{x} < X) \Rightarrow Q(\underline{x}))) \Rightarrow Q(X).$$

The $\Rightarrow$-r and $\wedge$-l rules of the sequent calculus lead to the discovery of the induction hypothesis:

$$Th_u, X \in \mathcal{T}, \forall \underline{x}\,((\underline{x} \in \mathcal{T} \wedge \underline{x} < X) \Rightarrow Q(\underline{x})) \vdash Q(X).$$

In deduction modulo, the induction hypothesis can now be internalized as a conditional equality denoted in general $\mathcal{RE}_{ind}(Q, <, \mathcal{T})(X)$ or shortly $\mathcal{RE}_{ind}(Q)$:

$$Q(\underline{x}) \text{ if } \underline{x} \in \mathcal{T} \wedge \underline{x} < X. \tag{1}$$

Note that because of its status of free fresh variable, X behaves like a constant, while $\underline{x}$ is universally quantified. Under these settings, we are left to derive the sequent

$$Th_u, X \in \tau \vdash_{\mathcal{RE}_{ind}(Q,<,\mathcal{T})(X)} Q(X)$$

in the sequent calculus modulo.

From now on, we instantiate $\mathcal{T}$ by the set of ground terms $\mathcal{T}(\Sigma)$, and $<$ by the proper part of a quasi simplification ordering (see e.g [WEC 92]) $\leqslant$ defined on the set of terms $\mathcal{T}(\Sigma, \mathcal{X})$. In order to compare n-tuples of terms, we use the standard extension on the Cartesian product $\leqslant_n$ of $\leqslant$: $\forall \overrightarrow{u}, \overrightarrow{v} \in \mathcal{T}(\Sigma, \mathcal{X})^n \quad \overrightarrow{u} \leqslant_n \overrightarrow{v} \Leftrightarrow (\forall i\ 1 \leq i \leq n \Rightarrow u_i \leqslant v_i)$ ($<_n$ denotes the proper part of $\leqslant_n$).
Also, as useful later, the quasi ordering $\leqslant_2$ can be used to orient equalities: $s \approx t \leqslant_2 s' \approx t'$ if $(s, t) \leqslant_2 (s', t')$.

When using such an ordering, the internalized induction hypothesis becomes:

$$\mathcal{RE}_{ind}(Q, <_n, \mathcal{T}(\Sigma)^n)(\overrightarrow{X}) : \quad (\overrightarrow{\underline{x}} \in \mathcal{T}(\Sigma)^n \wedge \overrightarrow{\underline{x}} <_n \overrightarrow{X}) \Rightarrow Q(\overrightarrow{\underline{x}})$$

where $\overrightarrow{X}$ is therefore the vector of free variables of $\mathcal{RE}_{ind}(Q, <_n, \mathcal{T}(\Sigma)^n)$. To make it precise and because only the free variables can be instantiated, we denote by $\mathcal{RE}_{ind}(Q)\sigma$ the instantiation by σ of the internalized induction hypothesis:

$$\mathcal{RE}_{ind}(Q)\sigma : \quad (\overrightarrow{\underline{x}} \in \mathcal{T}(\Sigma)^n \wedge \overrightarrow{\underline{x}} <_n \overrightarrow{X}\sigma) \Rightarrow Q(\overrightarrow{\underline{x}}).$$

5 A proof search system for induction modulo

The powerful principle of the approach used by systems like Spike [BOU 92] or RRL [KAP 95] is to allow application of rewrite rules of the theory at any position of the current goal, as well as application of induction hypotheses and current conjecture, provided that the applied formula is smaller in the Noetherian induction ordering than the current goal. When the ordering contains the relation induced by a terminating rewrite system, a smaller formula is obtained as soon as a rewrite step is performed. Moreover, in Spike for instance, the choice of the induction variables and instantiation schemas is done using pre-calculated induction positions and schemas called test-sets. In the approach described below, we show how to use narrowing to automatically and completely perform these choices, thanks to results of Section 3.

The proof search system IndNarrowModE for inductive proofs introduced in this section is based on narrowing and rewriting. The main rule, called **Induce**, performs the induction step. Its intuition is the following: in order to apply the induction hypothesis, one should decrease the size of the goal by rewriting it using a noetherian rewrite system. Whenever the goal does not rewrite, it should be first instantiated to be then rewritten, i.e. it should be narrowed. By expressing this in the sequent calculus modulo, we provide an explicit and constructive bridge between the rewrite-based implicit and explicit approaches of induction.

5.1 The proof search system IndNarrowModE

In order to provide the notational support for expressing our proof search methodology, we write sequents

$$\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q$$

where $\mathcal{RE}_1$, Γ_1, $\mathcal{RE}_2$, Γ_2 are such that:

- $\mathcal{RE}_1$ contains non conditional rewrite equalities or rules $l \to r$ of the user specification such that l is not a constructor term,
- Γ_1 contains other axioms of the user specification,
- $\mathcal{RE}_2$ contains the induction hypotheses and other required lemmas (in other words, RE_2 will collect the set of applied instantiations of the induction hypothesis)
- Γ_2 contains:
  - the Leibniz definition of equality: $L(\approx) : \forall x\ \forall y\ x \approx y \Rightarrow (\forall P\ P(x) \Rightarrow P(y))$;
  - crucial propositions for induction: the proposition $Noeth(<, \mathcal{T}(\Sigma))$ and the proposition NI defined above.
- Q is an equational goal.

In other words, Γ_1 is the deductive part of the user definitions, $\mathcal{RE}_1$ is their computational part; Γ_2 is the deductive part for other statements, $\mathcal{RE}_2$ is their computational part. The reader must bear in mind that $\Gamma_1 | \Gamma_2 \vdash_{\mathcal{RE}_1 | \mathcal{RE}_2} Q$ is a shorthand notation for sequent $\Gamma_1 \cup \Gamma_2, \overrightarrow{\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})} \vdash_{\mathcal{RE}_1 \cup \mathcal{RE}_2} Q$ in sequent calculus modulo. We may omit membership conditions for variables occurring in $\overrightarrow{\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})}$ when they are not involved in a deduction step. The distinction between $\Gamma_1, \mathcal{RE}_1$ and $\Gamma_2, \mathcal{RE}_2$ is needed because only $\mathcal{RE}_1$ will be used for narrowing. Moreover, we assume from now on, that Γ_1 contains a constructor preserving theory E, such that $(\mathcal{RE}_1, E)$ is terminating and sufficiently complete modulo E. Finally, we assume that $s' =_E s \leqslant t =_E t'$ implies $s' \leqslant t'$ (we say that $\leqslant$ is E-*compatible*).

Induce $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail$

$$\bullet_{\substack{l \to r \in \mathcal{RE}_1 \\ \sigma' \in CSUC_E(Q'_{|\omega'}, l)}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma' \cup \{\mathcal{RE}_{ind}(Q)\sigma'\}} (Q'[r]_{\omega'})\sigma'$$

if $Q' =_E Q$ and $\omega' \in DI(Q')$

Rewrite$_1$ $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q'$

if $Q \rightarrow_{\mathcal{RE}_1/E} Q'$

Rewrite$_2$ $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q'$

if $Q \rightarrow_{\mathcal{RE}_2/E} Q'$

Trivial $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} t \approx t' \rightarrowtail \diamond$

if $t =_E t'$

Refutation $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail$ Refutation

when no other rules can be applied

Fig. 3 The proof search system IndNarrowModE

Example 1 Assume that $\mathcal{RE}_1$ contains the rules of simple arithmetic given in Figure 1. $\mathcal{RE}_1$ is terminating and sufficiently complete modulo associativity and commutativity of the $*$ and $+$ operators (denoted $AC(+,*)$). Let $\Gamma_1 = AC(+,*)$, $\Gamma_2 = \{L(\approx), NI,\ Noeth(<, \mathcal{T}(\Sigma))\}$, and $Q = (x_1+x_2+x_3)*x_4 \approx x_1*x_4+x_2*x_4+x_3*x_4$. Then, we can consider the goal $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q$.

The proof search rules are presented in Figure 3. Sequents are gathered in a multiset structure modeled with the $\bullet$ operator that is an AC operator on sequents with $\diamond$ as neutral element. The rule **Induce** performs the induction step. It uses narrowing to choose both the induction variable(s) and the instantiation schema. Narrowing can be applied to any Q' E-equivalent to the current goal Q, always at defined-innermost positions. Furthermore, any application of the inference rule **Induce** provides a rule $\mathcal{RE}_{ind}(Q)\sigma'$, gathered in $\mathcal{RE}_2$. This rule is namely the *induction hypothesis* whose formal description was given in Section 4.2. **Trivial** eliminates a trivial equation, **Rewrite** (1 or 2) rewrites using a rule, an equation, or a smaller instance of a previous goal. **Rewrite** is duplicated because of the $\Gamma_1, \mathcal{RE}_1$ and $\Gamma_2, \mathcal{RE}_2$ distinction.
This inference rules set is generic and prepares to more operational versions tailored for AC and A-theories.

The sequent transformation described by the rules of IndNarrowModE in Figure 3 builds sequent derivations:

Definition 7 A *derivation* is any sequence of sequents:

$$\begin{aligned} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q &\rightarrowtail_{\text{IndNarrowModE}} \bullet_{i\in I_1}\ \Gamma_1^i|\Gamma_2^i \vdash_{\mathcal{RE}_1^i|\mathcal{RE}_2^i} Q^i \\ &\rightarrowtail_{\text{IndNarrowModE}} \cdots \bullet_{i\in I_k}\ \Gamma_1^i|\Gamma_2^i \vdash_{\mathcal{RE}_1^i|\mathcal{RE}_2^i} Q^i \cdots \end{aligned}$$

such that all rules **Rewrite** are applied with rewrite rules or equalities whose left-hand side is *strictly bigger* than its right-hand side (w.r.t. the ordering $<$).

When considering a derivation, it is important to assume that the proof search does not "forget" any subgoal. This is expressed formally by a fairness assumption.

Definition 8 A derivation of sequents

$$\begin{aligned}\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q &\rightarrowtail_{\mathsf{IndNarrowModE}} \bullet_{i\in I_1}\ \Gamma_1^i|\Gamma_2^i \vdash_{\mathcal{RE}_1^i|\mathcal{RE}_2^i} Q^i \\ &\rightarrowtail_{\mathsf{IndNarrowModE}} \cdots \bullet_{i\in I_k}\ \Gamma_1^i|\Gamma_2^i \vdash_{\mathcal{RE}_1^i|\mathcal{RE}_2^i} Q^i \ldots\end{aligned}$$

is *fair* if the set of persisting sequents $\cup_m \cap_{n\geq m} (\bullet_{i\in I_n}\ \Gamma_1^i|\Gamma_2^i \vdash_{\mathcal{RE}_1^i|\mathcal{RE}_2^i} Q^i)$ is empty.

5.2 Properties of IndNarrowModE

Soundness: Proving soundness amounts showing that for each rule of the proof search system IndNarrowModE of the form $S \rightarrowtail S'$, if S' is derivable in the sequent calculus modulo, then one can also build a proof of S. The main delicate point is to prove this result for the **Induce** rule, as stated in the next theorem.

Proposition 3 If the sequent

$$\Gamma_1|\Gamma_2,\ \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\cup\{\mathcal{RE}_{ind}(Q)\sigma'\}} (Q'[r]_{\omega'})\sigma'$$

is derivable in the sequent calculus modulo, where:

1. $Q =_E Q'$ and $\omega' \in DI(Q')$;
2. $l \rightarrow r \in \mathcal{RE}_1$ and $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$;
3. $\mathcal{RE}_2\sigma'$ is the rewrite system obtained by the replacement of each free variable x of any rewrite rule in $\mathcal{RE}_2$ by a corresponding $x\sigma'$;
4. $\overrightarrow{x_{\sigma'}} \in \mathcal{X}^{n_{\sigma'}}$ is the vector of free variables of $\mathcal{RE}_2\sigma' \cup \{Q\sigma'\}$;
5. $\overrightarrow{x} \in \mathcal{X}^n$ denotes the vector of free variables of $\mathcal{RE}_2 \cup \{Q\}$;

then, one can build a proof in the sequent calculus modulo of

$$\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$$

Proof. Sketch (for **Induce**): Let α be any ground instantiation of the variables in the set $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})$, $\alpha\downarrow$ one of its $\mathcal{RE}_1, E$-normalized form, and $V \subseteq \mathcal{X}$ such that $\mathcal{D}om(\alpha\downarrow) \subseteq V$. Since E is regular, $\mathcal{V}ar(Q) = \mathcal{V}ar(Q')$, thus $\alpha\downarrow$ is a ground $\mathcal{R}, E$-normalized instantiation of Q'. Proposition 2 and the assumptions, there exist a rewrite rule $l \rightarrow r \in \mathcal{RE}_1$, $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$, and a substitution μ', such that $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$, and $\sigma'\mu' =_E \alpha\downarrow [V]$. Now, since $E \subseteq \Gamma_1$, it is easy to make a proof in deduction modulo of the following sequent: $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} (\forall x\ x \in V \Rightarrow x\alpha \approx x\sigma'\mu')$ (1). Observe that sequent $\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\cup\{\mathcal{RE}_{ind}(Q)\sigma'\}} (Q'[r]_{\omega'})\sigma'$ is assumed, and since $l \rightarrow r \in \mathcal{RE}_1$, this leads to $\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\cup\{\mathcal{RE}_{ind}(Q)\sigma'\}} Q'\sigma'$, thus to $\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\cup\{\mathcal{RE}_{ind}(Q)\sigma'\}} Q\sigma'$ (since $E \subseteq \Gamma_1$ and $Q =_E Q'$). (2). Now, $x\alpha$ is ground for any $x \in \mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})$, thus $x\alpha\downarrow$ is also ground, which leads to $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\}) \subseteq \mathcal{D}om(\alpha\downarrow)$, hence, since $\mathcal{D}om(\alpha\downarrow) \subseteq V$, to $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\}) \subseteq V$. From the equivalence $\sigma'\mu' =_E \alpha\downarrow [V]$ and since E is regular, $x\sigma'\mu'$ is ground for any $x \in \mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})$. Thus, μ' is a ground instantiation of all variables in the set $\mathcal{V}ar(\mathcal{RE}_2\sigma' \cup \{Q\sigma'\})$ (3). We have shown in [NAH 07] that we obtain from (2), (3) and the assumption $\overrightarrow{x_{\sigma'}} = \overrightarrow{\mathcal{V}ar(\mathcal{RE}_2\sigma' \cup \{Q\sigma'\})}$ the

sequent $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\mu' \cup \{\mathcal{RE}_{ind}(Q)\sigma'\mu'\}} Q\sigma'\mu'$ (4). From sequents (1) and (4), and since $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\}) \subseteq V$, we obtain $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\alpha \cup \{\mathcal{RE}_{ind}(Q)\alpha\}} Q\alpha$ (5). Observe that sequent (5) is valid for any ground instantiation α of the variables in the set $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})$. Thus we have: $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2 \cup \{\mathcal{RE}_{ind}(Q)\}} Q$ (6) (see [NAH 07] for details). Putting together sequent (6) and assumption: $Noeth(<, \mathcal{T}(\Sigma)) \in \Gamma_2$, we have shown in [NAH 07] that this leads to: $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ and we are done. □

From Proposition 3, we deduce:

Theorem 1 *IndNarrowModE is sound.*

Refutational correctness: Proving refutational correctness amounts showing that for each rule of the proof search system IndNarrowModE of the form $S \rightarrowtail S'$, if S is derivable in the sequent calculus modulo, then one can also build a proof of S'. Again the main delicate point is for the **Induce** rule, and is stated as follows.

Proposition 4 If the sequent $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ where $\overrightarrow{x} \in \mathcal{X}^n$ is the vector of free variables of $\mathcal{RE}_2 \cup \{Q\}$, admits a proof in the sequent calculus modulo, then one can build a proof of:

$$\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma' \cup \{\mathcal{RE}_{ind}(Q,<)\sigma'\}} (Q'[r]_{\omega'})\sigma'$$

where $Q' =_E Q$, $l \to r \in \mathcal{RE}_1$, $\omega' \in DI(Q')$, $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$, and $\overrightarrow{x_{\sigma'}} \in \mathcal{X}^{n_{\sigma'}}$ is the vector of free variables of $\mathcal{RE}_2\sigma' \cup \{Q\sigma'\}$.

Proof. Sketch (for **Induce**): Let $l \to r \in \mathcal{RE}_1$, $\sigma' \in CSUC_E(Q'_{|\omega'}, l)$, and μ' be any ground instantiation of the variables in the set $\mathcal{V}ar(\mathcal{RE}_2\sigma' \cup \{Q\sigma'\})$. $\sigma'\mu'$ is a ground instantiation of the variables in the set $\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})$ (1). Now, recall that $\overrightarrow{x} = \overrightarrow{\mathcal{V}ar(\mathcal{RE}_2 \cup \{Q\})}$, and that sequent $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ is assumed. Put together with (1), this leads to sequent $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'\mu'} Q\sigma'\mu'$ (2) (see [NAH 07] for details).
Now, if $\overrightarrow{x_{\sigma'}} = \overrightarrow{\mathcal{V}ar(\mathcal{RE}_2\sigma' \cup \{Q\sigma'\})}$, and since sequent (2) is valid for *any* ground instantiation μ' of the variables in the set $\mathcal{V}ar(\mathcal{RE}_2\sigma' \cup \{Q\sigma'\})$, we have shown in [NAH 07] that we obtain $\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma'} Q\sigma'$. Since $Q' =_E Q$, $E \subseteq \Gamma_1$ and $l \to r \in \mathcal{RE}_1$, it is easy to show that sequent $\Gamma_1|\Gamma_2, \overrightarrow{x_{\sigma'}} \in \mathcal{T}(\Sigma)^{n_{\sigma'}} \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma' \cup \mathcal{RE}_{ind}(Q)\sigma'} Q'[r]_{\omega'}\sigma'$ follows. □

Refutational completeness: The idea of *Refutational completeness* is that the proof search system will stop in Refutation if the initial conjecture is not an inductive theorem. Proving refutational completeness is achieved thanks to the **Refutation** rule which applies when no other rule of IndNarrowModE can be applied.

Proposition 5 1. If $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \overset{*}{\rightarrowtail}$ Refutation then the sequent $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ has no proof in sequent calculus modulo, where $\overrightarrow{x} \in \mathcal{X}^n$ is the vector of free variables of $\mathcal{RE}_2 \cup \{Q\}$.

2. If the sequent $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ has no proof in sequent calculus modulo, all fair derivations starting with $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ terminate with Refutation.

Proof. Sketch:

1. Let us assume that $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail^{*}_{\mathsf{IndNarrowModE}} \mathsf{Refutation}$
   There exist contexts $\Gamma'_1,\ \Gamma'_2$, rewrite systems $\mathcal{RE}'_1,\ \mathcal{RE}'_2$ and an equational goal Q' such that:

$$\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail^{*}_{\mathsf{IndNarrowModE}} \Gamma'_1|\Gamma'_2 \vdash_{\mathcal{RE}'_1|\mathcal{RE}'_2} Q' \rightarrowtail_{\substack{\mathsf{IndNarrowModE}\\ \mathsf{Refutation}}}$$

   If Q' contains any defined symbol, then there exists a defined-innermost position ω' in $Dom(Q')$. Let $\alpha\downarrow$ be a ground $\mathcal{R}, E$-normalized substitution. By Proposition 2, the set $CSUC_E(Q'_{|\omega'}, l)$ contains at least one element, and **Induce** can be applied, which leads to a contradiction. Thus, the equality Q' does not contain any defined symbol. However, the rule **Trivial** cannot be applied either, thus we have $Q' = s \approx t$, with $s,\ t$ constructor terms that are not $E-$equivalent, and the sequent $\Gamma'_1|\Gamma'_2 \vdash_{\mathcal{RE}'_1|\mathcal{RE}'_2} Q'$ has no proof, since the constructors are assumed to be free and $\approx$ satisfies the axioms of equality. Therefore, by refutational correctness of $\mathsf{IndNarrowModE}$ $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ has no proof either.
2. Let us assume that $\Gamma_1|\Gamma_2, \overrightarrow{x} \in \mathcal{T}(\Sigma)^n \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q$ has no proof in sequent calculus modulo, and consider a fair derivation:

$$\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail \dots \tag{2}$$

   One can easily check ([NAH 07]) that there is at least one subgoal $\Gamma^i_1|\Gamma^i_2 \vdash_{\mathcal{RE}^i_1|\mathcal{RE}^i_2} Q^i$, and one ground substitution α^i, such that $\Gamma^i_1|\Gamma^i_2 \vdash_{\mathcal{RE}^i_1|\mathcal{RE}^i_2\alpha^i\cup\{\mathcal{RE}_{ind}(Q^i)\alpha^i\}} Q^i\alpha^i$ has no proof in sequent calculus modulo. Since $<_2$ is well-founded, these $Q^i\alpha^i$ have a minimal element, say $Q^m\alpha^m$. Consequently, there is a subgoal $\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q^m$ such that:

$$\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2\alpha^m\cup\{\mathcal{RE}_{ind}(Q^m)\alpha^m\}} Q^m\alpha^m \text{ has no proof in sequent calculus modulo} \tag{3}$$

Since the derivation is fair, an inference rule will be applied to the above subgoal.

- Case 1: $\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q^m \rightarrowtail_{\mathbf{Trivial}} \diamond$, then $Q^m = s^m \approx t^m$, with $s^m =_E t^m$, and this leads obviously to a contradiction with (3).
- Case 2: $\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q^m \rightarrowtail_{\mathbf{Refutation}} \mathbf{Refutation}$, then we are done.
- Case 3: $\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q^m \rightarrowtail_{\mathbf{Rewrite}} \Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q'^m$.
  Let us note first that $\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2} Q'^m$ is a subgoal in a derivation, second that $<$ is the proper part of a quasi simplification ordering E-compatible and third, that $Q'^m =_E Q^m$, we get $Q'^m\alpha^m <_2 Q^m\alpha^m$. Thus, by minimality of $Q^m\alpha^m$, the sequent:

$$\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2\alpha^m\cup\{\mathcal{RE}_{ind}(Q'^m)\alpha^m\}} Q'^m\alpha^m \tag{4}$$

  has a proof in the sequent calculus modulo. However, we have shown in [NAH 07] that the above sequent leads to:

$$\Gamma^m_1|\Gamma^m_2 \vdash_{\mathcal{RE}^m_1|\mathcal{RE}^m_2\alpha^m\cup\{\mathcal{RE}_{ind}(Q^m)\alpha^m\}} Q^m\alpha^m \tag{5}$$

  Finally, since (3) and (5) cannot hold together, we obtain a contradiction.

– Case 4: Assume that

$$\textbf{Induce } \Gamma_1^m|\Gamma_2^m \vdash_{\mathcal{RE}_1^m|\mathcal{RE}_2^m} Q^m \rightarrowtail \bullet_{\substack{l \to r \in \mathcal{RE}_1 \\ \sigma' \in CSUC_E(Q'^m_{|\omega'}, l)}} \Gamma_1^m|\Gamma_2^m \vdash_{\mathcal{RE}_1^m|\mathcal{RE}_2^m\sigma' \cup \{\mathcal{RE}_{ind}(Q^m)\sigma'\}} (Q'^m[r]_{\omega'})\sigma'$$
$$\text{with } Q'^m =_E Q^m \text{ and } \omega' \in DI(Q'^m)$$

Now, by Proposition 2, there is $\sigma'^m \in CSUC_E(Q'^m_{|\omega'}, l)$, and a ground substitution μ^m, such that the equivalence $x\alpha^m =_E x\sigma'^m\mu^m$ holds for any $x \in \mathcal{V}ar(\mathcal{RE}_2^m \cup \{Q'^m\})$. Observe that $\Gamma_1^m|\Gamma_2^m \vdash_{\mathcal{RE}_1^m|\mathcal{RE}_2^m\sigma'^m \cup \{\mathcal{RE}_{ind}(Q^m)\sigma'^m\}} (Q'^m[r]_\omega)\sigma'^m$ is a subgoal, and $(Q'^m[r]_\omega)\sigma'^m\mu^m <_2 Q'^m\alpha^m$. Thus, by minimality of $Q^m\alpha^m$, the sequent:

$$\Gamma_1^m|\Gamma_2^m \vdash_{\mathcal{RE}_1^m|\mathcal{RE}_2^m\sigma'^m\mu^m \cup \{\mathcal{RE}_{ind}(Q^m)\sigma'^m\mu^m, \mathcal{RE}_{ind}((Q'^m[r]_\omega)\sigma'^m)\mu^m\}} (Q'^m[r]_\omega)\sigma'^m\mu^m$$

has a proof in the sequent calculus modulo. However, we have shown in [NAH 07]) that the above sequent leads to:

$$\Gamma_1^m|\Gamma_2^m \vdash_{\mathcal{RE}_1^m|\mathcal{RE}_2^m\alpha^m \cup \{\mathcal{RE}_{ind}(Q^m)\alpha^m\}} Q^m\alpha^m \tag{6}$$

Finally, since (3) and (6) cannot hold together, we obtain a contradiction. □

The above proof provides a hint to prove that a conjecture is not an inductive theorem: the proof search will probably stop sooner if we focus on subgoals whose ground instances are minimal at each inference step (w.r.t. the ordering $<_2$).

From Propositions 4 and 5, we obtain:

Theorem 2 *IndNarrowModE is refutationally correct and complete.*

6 Induction modulo *AC* and *A*

The general IndNarrowModE proof search system is indeed working directly on equivalence classes modulo E, a situation not directly implementable for most theories E. To focus on more operational proof search systems, we focus in this section on the case of associative-commutative or associative theories. We introduce two proof search systems IndNarrowModAC and IndNarrowModA as special instances of IndNarrowModE with specific improvements and illustrating examples. Soundness and refutational correctness and completeness of these systems will be consequences of the properties of IndNarrowModE.

6.1 More about flattened terms

In associative and associative-commutative theories, equivalence classes of terms are often represented by flattened terms. We refer for the basic definitions and results about positions and subterms to [MAR 93]. Intuitively flattening a term amounts to recursively replace $f(f(s,t),u)$ or $f(s,f(t,u))$ by $f(s,t,u)$ if f is an associative symbol. This is the key point bridging the proof search systems IndNarrowModAC and IndNarrowModA on the one hand, and IndNarrowModE on the other hand.

From now on, we assume that some function symbols in a subset $\mathcal{V}$ of Σ may have an *unbounded arity*. First, let us introduce the following definition:

Definition 9 For any $t \in \mathcal{T}(\Sigma, \mathcal{X})$, the *flattening* of t, denoted $\overline{t}$, is the normal form of t wrt the set of rewrite rules:

$$fx_1 \dots x_n(fy_1 \dots y_m)z_1 \dots z_r \to fx_1 \dots x_n y_1 \dots y_m z_1 \dots z_r$$

for any $f \in \mathcal{V}$, $m,\ n,\ r \in \mathbb{N}$ such that $n \geq 1$ or $r \geq 1$, and $m \geq 2$.
Hence, $\overline{t}$ is called *flattened term*.

Let $A(\mathcal{V}) = \{f(fxy)z \approx fx(fyz) \mid f \in \mathcal{V}\}$ and $AC(\mathcal{V}) = A(\mathcal{V}) \cup \{fxy \approx fyx \mid f \in \mathcal{V}\}$. In the following, we assume that all symbols in $\mathcal{V}$ are defined symbols, that constructor symbols do *not* have unbounded arities, and that $+$ and $*$ denote symbols with unbounded arity.

[MAR 93] defines a transformation which associates to each position in a given term t a position in the flattening $\overline{t}$ of t, also called the *flattening* of this position. However, a position in $\overline{t}$ is not always the flattening of some position in t, and this led us to introduce the following definition:

Definition 10 For a flattened term $\overline{s}$, a position $\omega \in \mathcal{D}om(\overline{s})$ is *flattened* if there exist $i, k \in \mathbb{N}$, and a string ω_0, s.t. $\omega = \omega_0.i$ or $\omega = \omega_0.\{i, i+1, \dots, i+k\}$.

As an example, $\overline{t} = succ(+(a,b,c,d))$ is the flattening of $t = succ(((a+b)+c)+d)$. The position $1.\{2,3\}$ denotes the subterm $+(b,c)$ and is flattened with respect to the definition above, whereas $1.\{1,3,4\}$ denotes $+(a,c,d)$ but is not flattened.

The above flattened positions are precisely the flattening of positions in the sense of [MAR 93], see [NAH 07] for the proof. To define a rewrite relation on the set of flattened terms, the notion of replacement has to be generalized:

Definition 11 Given two flattened terms $\overline{s} = f\overline{s_1} \dots \overline{s_n}$, $\overline{t}$, and a position $\omega \in \overline{s}$, the replacement by $\overline{t}$ in $\overline{s}$ at the position ω is inductively defined by:
- $\overline{s}[\overline{t}]_\varepsilon = \overline{t}$
- If $\omega \in \{1, \dots, n\}$
  - Case 1: there exist $i,\ k \in \mathbb{N}$, such that $\omega = \{i, i+1, \dots, i+k\}$.
    $\overline{s}[\overline{t}]_\omega = \overline{f\overline{s_1} \dots \overline{s_{i-1}}\ \overline{t}\ \overline{s_{i+k+1}} \dots \overline{s_n}}$
  - Case 2: otherwise, let $\{i_1, \dots, i_k\} = \{1, \dots, n\} - \omega$.
    $\overline{s}[\overline{t}]_\omega = \overline{f\overline{s_{i_1}} \dots \overline{s_{i_k}}\ \overline{t}}$.
- $\overline{s}[\overline{t}]_{i.\omega_i} = \overline{f\overline{s_1} \dots \overline{s_i}[\overline{t}]_{\omega_i} \dots \overline{s_n}}$.

Take for instance $\overline{s} = succ(+(a,b,c,d))$ and $\overline{t} = +(e,f,g)$. Then

$$\begin{aligned}
\overline{s}[\overline{t}]_{1.\{2,3\}} &= succ(+(a,b,c,d))[+(e,f,g)]_{1.\{2,3\}} \\
&= succ(\overline{+(a,b,c,d)[+(e,f,g)]_{\{2,3\}}}) \\
&= succ(\overline{+(a,+(e,f,g),d)}) \\
&= succ(+(a,e,f,g,d))
\end{aligned}$$

$$\begin{aligned}
\overline{s}[\overline{t}]_{1.\{1,3\}} &= succ(+(a,b,c,d))[+(e,f,g)]_{1.\{1,3\}} \\
&= succ(\overline{+(a,b,c,d)[+(e,f,g)]_{\{1,3\}}}) \\
&= succ(\overline{+(b,d,+(e,f,g))}) \\
&= succ(+(b,d,e,f,g))
\end{aligned}$$

A rewrite relation on the set of flattened terms is now defined as follows:

Definition 12 Given a rewrite system $\mathcal{R}$, $\overline{s}, \overline{t}$ two flattened terms. $\overline{s} \to_{\overline{\mathcal{R}}} \overline{t}$ if there is a rule $c \Rightarrow l \to r \in \mathcal{R}$, a flattened position $\omega \in \mathcal{D}om(\overline{s})$ and a substitution σ such that:

InduceAC $\quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail$

$\bullet_{\substack{l \to r \in \mathcal{RE}_1 \\ \sigma \in CSUC_{AC}(\overline{Q}_{|\omega}, l)}} \quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma \cup \{\mathcal{RE}_{ind}(Q)\sigma\}} (\overline{Q}[\overline{r}]_\omega)\sigma$

if $\omega \in DI(\overline{Q})$

Rewrite$_1$AC $\quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q'}$

if $\overline{Q} \rightarrow_{\overline{\mathcal{RE}_1}/\equiv_p} \overline{Q'}$

Rewrite$_2$AC $\quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q'}$

if $\overline{Q} \rightarrow_{\overline{\mathcal{RE}_2}/\equiv_p} \overline{Q'}$

TrivialAC $\quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{t} \approx \overline{t'} \rightarrowtail \diamond$

if $\overline{t} \equiv_p \overline{t'}$

RefutationAC $\quad \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail$ Refutation

when no other rules can be applied

Fig. 4 The proof search system IndNarrowModAC

– $\overline{s}_{|\omega} = \overline{l\sigma}$, $\overline{t} = \overline{s}[\overline{r\sigma}]_\omega$
– and the condition $c\sigma$ is true.

For example, given $\mathcal{R} = \{x + 0 \to x\}$, $succ(+(a, b, 0, d)) \rightarrow_{\overline{\mathcal{R}}} succ(+(a, b, d))$.

If $\equiv_p$ denotes the classical equivalence induced on the set of flattened terms by permutating the arguments of symbols in $\mathcal{V}$, let us consider the extension $\overline{\mathcal{R}}/\equiv_p$ of $\overline{\mathcal{R}}$ on the set of $\equiv_p$-equivalences. As previously, in order to perform induction by narrowing at *defined-innermost* positions, we must define such positions for flattened terms:

Definition 13 For any $s \in \mathcal{T}(\Sigma, \mathcal{X})$, and for any $\omega \in \mathcal{D}om(\overline{s})$, the position ω is called *defined-innermost* whenever there exist $f \in \Sigma$ and terms $s_1, \ldots, s_n \in \mathcal{T}(\mathcal{C}, \mathcal{X})$, such that $\overline{s}_{|\omega} = f\overline{s_1} \ldots \overline{s_n}$, and moreover $n = 2$ if $f \in \mathcal{V}$.

Intuitively, the position ω in $\overline{s}$ is defined-innermost when $\overline{s}_{|\omega}$ coincides with its flattened form.

For example, if $+$ is a defined symbol, then the position 2.1 is defined-innermost in $f(+(1, 2, 3), g(+(4, 5)))$ while the position 1 is not.

6.2 The proof search systems IndNarrowModAC and IndNarrowModA

The specific proof search systems IndNarrowModAC and IndNarrowModA are respectively given in Figure 4 and Figure 5.

Soundness, refutational correctness and completeness of IndNarrowModAC and IndNarrowModA are consequences of the following proposition that states a correspondence between a deduction on a goal Q using IndNarrowModE and a deduction on the corresponding flattened goal using IndNarrowModAC or IndNarrowModA.

Theorem 3 *Let $E = AC(\mathcal{V})$ (resp.$E = A(\mathcal{V})$).*

InduceA $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail$

$\bullet_{\substack{l \to r \in \mathcal{RE}_1 \\ \sigma \in CSUC_A(\overline{Q}_{|\omega}, l)}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2\sigma \cup \{\mathcal{RE}_{ind}(Q)\sigma\}} (\overline{Q}[\overline{r}]\omega)\sigma$

if $\omega \in DI(\overline{Q})$ and ω flattened.

Rewrite$_1$A $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q'}$

if $\overline{Q} \rightarrow_{\overline{\mathcal{RE}_1}} \overline{Q'}$

Rewrite$_2$A $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q'}$

if $\overline{Q} \rightarrow_{\overline{\mathcal{RE}_2}} \overline{Q'}$

TrivialA $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{t} \approx \overline{t'} \rightarrowtail \diamond$

if $\overline{t} = \overline{t'}$

RefutationA $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail$ Refutation

when no other rules can be applied

Fig. 5 The proof search system IndNarrowModA

1. *If* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{\mathsf{IndNarrowModE}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} R$, *there exists* R' *such that* $R' =_{AC} R$, *and* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathsf{IndNarrowModAC}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \overline{R'}$. *(resp.* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathsf{IndNarrowModA}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \overline{R}$ *).*

2. *If* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathsf{IndNarrowModAC}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \overline{R}$ *(resp.* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathsf{IndNarrowModA}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} \overline{R}$*), there exists* R' *such that* $R' =_{AC} R$ *(resp.* $R' =_A R$*), and* $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{\mathsf{IndNarrowModE}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}'_2} R'$

Proof. Sketch: We only consider here the case $E = AC(\mathcal{V})$, the case $E = A(\mathcal{V})$ being very similar. Let us focus on the most delicate point, which is the case of the rule **Induce** and consider the two cases.

1. $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{\mathbf{Induce}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2 \cup \{\mathcal{RE}_{ind}(Q)\sigma'\}} (Q'[r]_{\omega'})\sigma'$. By definition, there exists $l \to r \in \mathcal{RE}_1$ such that $\sigma' \in CSUC_{AC}(Q'_{|\omega'}, l)$. Thus, we have $\overline{Q'_{|\omega'}\sigma'} \equiv_p \overline{l\sigma'}$, which leads to $\overline{Q'_{|\omega'}}\sigma' \equiv_p \overline{l\sigma'}$ (1), since σ' is a constructor substitution and it is assumed that constructor symbols do not have unbounded arities. Now, if $flat_{Q'}(\omega')$ denotes the flattening of the position ω', we have $\overline{Q'_{|\omega'}} = \overline{Q'}_{|flat_{Q'}(\omega')}$ (2) ([MAR 93]).
   From the assumption $\omega' \in DI(Q')$ and the above equality, it is easy to see that $flat_{Q'}(\omega') \in DI(\overline{Q'})$. And, since σ' is a constructor substitution, $\overline{Q'}_{|flat_{Q'}(\omega')}\sigma'$ can be seen as a term which is equal to its own flattening (3). From (1), (2) and (3), σ' is an AC-unifier of $\overline{Q'}_{|flat_{Q'}(\omega')}$ and l. Furthermore, since $\sigma' \in CSUC_{AC}(Q'_{|\omega'}, l)$, one can easily show that we have $\sigma' \in CSUC_{AC}(\overline{Q'}_{|flat_{Q'}(\omega')}, l)$. By definition of **Induce**, $Q' =_{AC} Q$, and we have shown in [NAH 07] that there exists $\omega \in DI(\overline{Q})$ such that $\overline{Q'}_{|flat_{Q'}(\omega')} \equiv_p \overline{Q}_{|\omega}$ and $\overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')} \equiv_p \overline{Q}[\overline{r}]_{\omega}$. Therefore $\sigma' \in CSUC_{AC}(\overline{Q}_{|\omega}, l)$ and we have the inference step:

$\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathbf{InduceAC}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2 \cup \mathcal{RE}_{ind}(Q)\sigma'} (\overline{Q}[\overline{r}]_\omega)\sigma'$ (4). Let us recall that $\overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')} \equiv_p \overline{Q}[\overline{r}]_\omega$ (5). Furthermore, we have shown in [NAH 07] the equality $\overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')} = \overline{Q'[r]_{\omega'}}$ (6). From (4), (5) and (6), we obtain $(\overline{Q}[\overline{r}]_\omega)\sigma' \equiv_p \overline{Q'[r]_{\omega'}\sigma'}$ and the conclusion follows.

2. $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} \overline{Q} \rightarrowtail_{\mathbf{InduceAC}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2 \cup \{RE_{ind}(Q)\sigma\}} (\overline{Q}[\overline{r}]_\omega)\sigma$. By definition of the rule **InduceAC**, there exist $\omega \in DI(\overline{Q})$ and $l \rightarrow r \in \mathcal{RE}_1$, such that $\sigma \in CSUC_{AC}(\overline{Q}_{|\omega}, l)$. Then, we have shown in [NAH 07] that there exist an equality Q' and a position $\omega' \in \mathcal{D}om(Q')$, such that $Q' =_{AC} Q$, $\overline{Q}_{|\omega} = \overline{Q'}_{|flat_{Q'}(\omega')}$, and $\overline{Q}[\overline{r}]_\omega \equiv_p \overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')}$ (1). Therefore, we have $\overline{Q'}_{|flat_{Q'}(\omega')}\sigma =_{AC} l\sigma$, and since $\overline{Q'_{|\omega'}} = \overline{Q'}_{|flat_{Q'}(\omega')}$ ([MAR 93]), this leads to $\overline{Q'_{|\omega'}}\sigma =_{AC} l\sigma$. Since $\omega \in DI(\overline{Q})$, it is not difficult to show that $\omega' \in DI(Q')$, thus $\overline{Q'_{|\omega'}} = Q'_{|\omega'}$. Replacing in the previous equality, we obtain $Q'_{|\omega'}\sigma =_{AC} l\sigma$. Thus, σ is an AC-unifier of $Q'_{|\omega'}$ and l. Furthermore, one can easily show that σ is in $CSUC_{AC}(Q'_{|\omega'}, l)$. Hence we have the inference step:
$\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2} Q \rightarrowtail_{\mathbf{Induce}} \Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\mathcal{RE}_2 \cup \{\mathcal{RE}_{ind}(Q)\sigma\}} (Q'[r]_{\omega'})\sigma$ (2). Now, let us observe that we have $\overline{(Q'[r]_{\omega'})\sigma} = \overline{(Q'[r]_{\omega'})}\sigma$. Furthermore, since $\overline{Q'[r]_{\omega'}} = \overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')}$ ([MAR 93]), we obtain $\overline{(Q'[r]_{\omega'})\sigma} = (\overline{Q'}[\overline{r}]_{flat_{Q'}(\omega')})\sigma$. Thus, by (1), $\overline{(Q'[r]_{\omega'})\sigma} \equiv_p (\overline{Q}[\overline{r}]_\omega)\sigma$, and linking with (2), we are done.

□

6.3 Two simple examples

In order to get a better intuition on the way these sets of rules are working, let us look at two examples. In the following, we always refer to the specification and the set of rewrite rules given in Figure 1. The first example of proof uses AC properties of $+$ and $*$ induction modulo AC, and the second one uses the same rules but just associativity of these two symbols.

The case of associativity and commutativity (AC): Recall the context of Example 1: $\mathcal{RE}_1$ is assumed to contain the rules of simple arithmetic given in Figure 1, $\Gamma_1 = AC(+, *)$, $\Gamma_2 = \{L(\approx), NI,\ Noeth(<, \mathcal{T}(\Sigma))\}$, and $Q = (x_1 + x_2 + x_3) * x_4 \approx x_1 * x_4 + x_2 * x_4 + x_3 * x_4$.
Let us consider the following sequent: $\Gamma_1|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} Q$ and first apply the rule **InduceAC**. The innermost positions in Q are 1.1.$\{1, 2\}$, 1.1.$\{1, 3\}$, 1.1.$\{2, 3\}$, 2.1, 2.2 and 2.3. Since the goal remains equivalent by permutation of the variables x_1, x_2 and x_3, only two possibilities remain: narrowing at a position where the symbol $+$ occurs, or where the symbol $*$ occurs. Since the last choice creates more reductions than the first one, we arbitrarily choose to narrow at the position 2.1 of the goal. Therefore, we must compute the set $CSUC_{AC}(x_1 * x_4, l)$ for any rewrite rule $l \rightarrow r$ of $\mathcal{RE}_1$. This

restricts to rules such that $l(\varepsilon) = *$, and we obtain:

l	$CSUC_{AC}(x_1 * x_4, l)$
$x * 0$	$\sigma_1 = \{x_1 \to y_1; x \to y_1; x_4 \to 0\}$ $\sigma_2 = \{x_1 \to 0; x \to y_4; x_4 \to y_4\}$
$x * s(y)$	$\sigma_3 = \{x_1 \to y_1; x \to y_1; y \to y_4; x_4 \to s(y_4)\}$ $\sigma_4 = \{x_1 \to s(y_1); x \to y_4; y \to y_1; x_4 \to y_4\}$

After normalization, this leads us to prove the four sequents:

$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_1}$	$0 \approx 0$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2}$	$(x_2 + x_3) * y_4 \approx x_2 * y_4 + x_3 * y_4$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_3}$	$(y_1 + x_2 + x_3) * y_4 + y_1 + x_2 + x_3$ $\approx y_1 * y_4 + y_1 + x_2 * y_4 + x_2 + x_3 * y_4 + x_3$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_4}$	$(y_1 + x_2 + x_3) * y_4 + y_4 \approx y_1 * y_4 + y_4 + x_2 * y_4 + x_3 * y_4$

Trivial gets rid of the first one. Since $(y_1, x_2, x_3, y_4) <_4 (y_1, x_2, x_3, s(y_4))$, the induction hypothesis can be applied on the third one, and since $(y_1, x_2, x_3, y_4) <_4 (s(y_1), x_2, x_3, y_4)$, it can be applied on the fourth one too. Hence, the inference rule **Rewrite**$_2$ rewrites the third goal with the induction hypothesis $\mathcal{RE}_{ind}(Q)\sigma_3$, and the fourth goal with the induction hypothesis $\mathcal{RE}_{ind}(Q)\sigma_4$. Thus we get:

$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2}$	$(x_2 + x_3) * y_4 \approx x_2 * y_4 + x_3 * y_4$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_3}$	$y_1 * y_4 + x_2 * y_4 + x_3 * y_4 + y_1 + x_2 + x_3$ $\approx y_1 * y_4 + y_1 + x_2 * y_4 + x_2 + x_3 * y_4 + x_3$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_4}$	$y_1 * y_4 + x_2 * y_4 + x_3 * y_4 + y_4$ $\approx y_1 * y_4 + y_4 + x_2 * y_4 + x_3 * y_4$

Trivial gets rid of the two last subgoals. The application of **Induce** to the first one at position 2.1 generates four subgoals. **Trivial** gets rid of the two first ones, the application of **Rewrite**$_2$ to the last ones creates two new subgoals which are trivial and we are done.

The case of Associativity: Assume that $\mathcal{RE}_1$ contains the rules of simple arithmetic given in Figure 1. $\mathcal{RE}_1$ is terminating and sufficiently complete modulo associativity of the $*$ and $+$ operators (denoted $A(+, *)$) Let us prove that distributivity of $*$ over $+$ is an inductive theorem.
Let $\Gamma_1 = A(+, *)$, $\Gamma_2 = Th_{\approx} \cup \{NI,\ Noeth(<_3, \mathcal{T}(\Sigma)^3)\}$, and $Q = x_1 * (x_2 + x_3) \approx x_1 * x_2 + x_1 * x_3$. Let us start from the sequent: $\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \emptyset} Q$.
We can apply **InduceA** at the innermost positions 1.2, 2.1 and 2.2 in Q and Theorem 3 ensures that each of these choices is correct. Since narrowing at position 2.1 creates less further reductions than the ones at positions 1.2 or 2.2, we choose to narrow at the position 2.2 of the goal. Thus, we need to compute $CSUC_A(x_1 * x_3, l)$ for any rewrite rule $l \to r$ of $\mathcal{RE}_1$. This restricts to rules such that $l(\varepsilon) = *$, and we obtain:

l	$CSUC_A(x_1 * x_3, l)$
$x * 0$	$\sigma_1 = \{x_1 \to y_1; x \to y_1; x_3 \to 0\}$
$x * s(y)$	$\sigma_2 = \{x_1 \to y_1; x \to y_1; y \to y_3; x_3 \to s(y_3)\}$

After normalization, we obtain the subgoals:

$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_1} y_1 * x_2 \approx y_1 * x_2$
$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2} y_1 * (x_2 + y_3) + y_1 \approx y_1 * x_2 + y_1 * y_3 + y_1$

Trivial gets rid of the first subgoal. Since $(y_1, x_2, y_3) <_3 (y_1, x_2, s(y_3))$, **Rewrite$_2$A** can be applied on the second one. Hence, we get:

$\Gamma_1 \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2} y_1 * x_2 + y_1 * y_3 + y_1 \approx y_1 * x_2 + y_1 * y_3 + y_1$

and **Trivial** gets rid of this last subgoal.

6.4 A proof by refutation example

– **Sorts:** nat, list;
– **constructors:** $0 : \rightarrow nat$ $\quad s : nat \rightarrow nat$ $\quad :: \; : \; nat \times list \rightarrow list$
– **defined functions:** $<>$: $list \times list \rightarrow list$
– **rules**:

$$< nil, \bar{l} > \; \rightarrow \; \bar{l} \qquad < x :: \bar{l}, \overline{l'} > \; \rightarrow \; x ::< \bar{l}, \overline{l'} >$$

Fig. 6 Simple lists

Assume that $\mathcal{RE}_1$ contains the rules of **Simple lists** given in Figure 6. $\mathcal{RE}_1$ is terminating and sufficiently complete modulo associativity of the $<>$ operator (denoted $A(<>)$). Let $\Gamma_1 = A(<>)$, $\Gamma_2 = \{L(\approx), NI,\ Noeth(<, \mathcal{T}(\Sigma))\}$, and $Q =< L, M >\approx< M, L >$. Then, we can consider the goal:

Q	$\emptyset, A(<>) \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \emptyset} < L, M > \approx < M, L >$

– Let us apply **InduceA** at position 1.

l	$CSUC_{A(<>)}(< L, M >, l)$
$< nil, \bar{l} >$	$\sigma_1 = \{L \rightarrow nil; M \rightarrow M_1; \bar{l} \rightarrow M_1\}$
$< x :: \bar{l}, \overline{l'} >$	$\sigma_2 = \{L \rightarrow X_1 :: L_1; M \rightarrow M_1$ $x \rightarrow X_1; \bar{l} \rightarrow L_1; \overline{l'} \rightarrow M_1\}$

– After normalization, we obtain the subgoals:

Q_1	$\emptyset, A(<>) \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_1} M_1 \approx < M_1, nil >$
Q_2	$\emptyset, A(<>) \vert \Gamma_2 \vdash_{\mathcal{RE}_1 \vert \mathcal{RE}_{ind}(Q)\sigma_2} X_1 ::< L_1, M_1 > \approx < M_1, X_1 :: L_1 >$

– Let us apply **InduceA** at position 2 to the first subgoal Q_1:

l	$CSUC_{A(<>)}(< M_1, nil >, l)$
$< nil, \bar{l} >$	$\sigma_3 = \{M_1 \rightarrow nil; \bar{l} \rightarrow nil\}$
$< x :: \bar{l}, \overline{l'} >$	$\sigma_4 = \{M_1 \rightarrow X_2 :: M_2$ $x \rightarrow X_2; \bar{l} \rightarrow M_2; \overline{l'} \rightarrow nil\}$

– After normalization, we obtain the subgoals:

Q_3	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_1\sigma_3,\mathcal{RE}_{ind}(Q_1)\sigma_3}$ $nil \approx nil$
Q_4	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_1\sigma_4,\mathcal{RE}_{ind}(Q_1)\sigma_4}$ $X_2 :: M_2 \approx X_2 ::< M_2, nil >$

– **TrivialA** gets rid of subgoal Q_3.
– Since $M_2 < X_2 :: M_2$ ($<$ is assumed to be any quasi simplification ordering showing termination of the specification **Simple lists**), the inference rule **Rewrite**$_2$ can be applied to Q_4 with $\mathcal{RE}_{ind}(Q_1)\sigma_4$, we obtain:

Q'_4	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2,\mathcal{RE}_{ind}(Q_1)\sigma_4}$ $X_2 :: M_2 \approx X_2 :: M_2$

– **TrivialA** gets rid of subgoal Q'_4.

– Let us apply **InduceA** at position 1.2 of subgoal Q_2.

l	$CSUC_{A(<>)}(< L_1, M_1 >, l)$
$< nil, \bar{l} >$	$\sigma_5 = \{L_1 \to nil; M_1 \to M_2; \bar{l} \to M_2\}$
$< x :: \bar{l}, \overline{l'} >$	$\sigma_6 = \{L_1 \to X_2 :: L_2; M_1 \to M_2$ $x \to X_2; \bar{l} \to L_2; \overline{l'} \to M_2\}$

– After normalization, we obtain the subgoals:

Q_5	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_5,\mathcal{RE}_{ind}(Q_2)\sigma_5}$ $X_1 :: M_2 \approx< M_2, X_1 :: nil >$
Q_6	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_6,\mathcal{RE}_{ind}(Q_2)\sigma_6}$ $X_1 :: (X_2 ::< L_2, M_2 >) \approx$ $< M_2, X_1 :: (X_2 :: L_2) >$

– Let us apply **InduceA** at position 2 of subgoal Q_5.

l	$CSUC_{A(<>)}(< M_2, X_1 :: nil >, l)$
$< nil, \bar{l} >$	$\sigma_7 = \{M_2 \to nil; X_1 \to X_2; \bar{l} \to X_2 :: nil\}$
$< x :: \bar{l}, \overline{l'} >$	$\sigma_8 = \{M_2 \to X_3 :: L_3; X_1 \to X_2$ $x \to X_3; \bar{l} \to L_3; \overline{l'} \to X_2 :: nil\}$

– After normalization, we obtain the subgoals:

Q_7	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_5\sigma_7,\mathcal{RE}_{ind}(Q_2)\sigma_5\sigma_7,\mathcal{RE}_{ind}(Q_5)\sigma_7}$ $X_2 :: nil \approx X_2 :: nil$
Q_8	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_6\sigma_8,\mathcal{RE}_{ind}(Q_2)\sigma_6\sigma_8,\mathcal{RE}_{ind}(Q_5)\sigma_8}$ $X_2 :: (X_3 :: L_3) \approx X_3 ::< L_3, X_2 :: nil >$

– **TrivialA** gets rid of subgoal Q_7.

– Let us apply **InduceA** at position 2.2 of subgoal Q_8.

l	$CSUC_{A(<>)}(< L_3, X_2 :: nil >, l)$
$< nil, \bar{l} >$	$\sigma_9 = \{L_3 \to nil; X_2 \to X_4; \bar{l} \to X_4 :: nil\}$
$< x :: \bar{l}, \overline{l'} >$	$\sigma_{10} = \{L_3 \to X_5 :: L_5; X_2 \to X_4$ $x \to X_5; \bar{l} \to L_5; \overline{l'} \to X_4 :: nil\}$

– After normalization, we obtain the subgoals:

Q_9	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_6\sigma_8\sigma_9,\mathcal{RE}_{ind}(Q_2)\sigma_6\sigma_8\sigma_9,\mathcal{RE}_{ind}(Q_5)\sigma_8\sigma_9,\mathcal{RE}_{ind}(Q_8)\sigma_9}$ $X_4 :: (X_3 :: nil) \approx X_3 :: (X_4 :: nil)$
Q_{10}	$\emptyset, A(<>)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2\sigma_6\sigma_8\sigma_{10},\mathcal{RE}_{ind}(Q_2)\sigma_6\sigma_8\sigma_{10},\mathcal{RE}_{ind}(Q_5)\sigma_8\sigma_{10},\mathcal{RE}_{ind}(Q_8)\sigma_{10}}$ $X_4 :: (X_3 :: (X_5 :: L_5)) \approx X_3 :: (X_5 ::< L_5, X_4 :: nil >)$

– **Refutation** applies to subgoal Q_9.

Thus, by theorem 5, the sequent:

$$(L, M) \in \mathcal{T}(\Sigma)^2, A(<>)|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} < L, M > \approx < M, L >$$

has no proof in the sequent calculus modulo.
Notice that in the above proof, all rules **Rewrite** have been applied with rewrite rules or equalities whose left-hand side were *strictly bigger* than their right-hand side, and that we have selected the "simplest" goal at each inference step.

7 Prototype and computer experiments

One major advantage of the semi-decision procedure presented in this paper is that there is a clear correspondence between each proof-search step and the structure of the goal's actual proof. Indeed, the proof of Theorem 3 being constructive, it is virtually possible to extract a proof in deduction modulo from each successful instance of the IndNarrow procedure. We present now what we have achieved so far to reach this goal.

7.1 Lemuridae

Lemuridae is a prototype proof assistant for Superdeduction Modulo [BRA 07, HOU 08] in the framework of sequent calculus. It contains in particular classical sequent calculus modulo and can therefore benefit from IndNarrow. Apart from allowing the user to build proofs in deduction modulo, it features a proof-term language based on Urban's language for sequent calculus [URB 01] and offers some automatic support for defining inductive types using an impredicative encoding [ALL 08], supported by a first-order encoding of higher-order logic through the theory of classes [KIR 07]. The expression of higher-order propositions therefore differs from the one chosen for IndNarrow, i.e. HOL-$\lambda\sigma$. Translating the proof of soundness from one system to another does not however raise major difficulties, since the higher-order part of the potentially extracted proofs should be confined to the formulation of the noetherian induction principle.
Lemuridae is written in Tom, a language that adds constructs inspired from rewriting theory to general-purpose languages, like Java or C. It offers an efficient term structure generator, associative pattern-matching and strategic programming. While associative pattern matching eases the development of Lemuridae's typechecker, the strategy language is particularly well-suited for expressing tactics and tacticals of the prover.

7.2 Prototype implementation of IndNarrow

A first implementation of IndNarrow has been written to give the algorithm a try before implementing it in Lemuridae or in other proof assistants. The program is very short - only 500 lines long - and behaves surprisingly well regarding some naive choices that have been made throughout the code. There are indeed some sources of nondeterminism we had to tackle, as discussed below. It is written in OCaml but reuses some Tom idioms in the expression of rewriting strategies, as well as in the idea of reifying the notion of position. This allows easily describing complex traversal strategies by means of some basic ones. For example, the `topdown` strategy is here defined using the fixpoint combinator `mu`, the sequence strategy `seq` and the strategy `all` that applies `x` to all the children of the current node.

```
let rec topdown s = mu (fun x -> seq s (all x))
```

What is original compared to strategic programming languages like Stratego [VIS 01] is that the strategy `s` always not only receives the current subterm as an argument, but also the current position. This allows for the concise expression of parts of the code that explicitly rely on positions.

This implementation work has pointed out the nondeterministic parts of IndNarrow. There are indeed four of them.

Rules application order. The algorithm does not specify any strategy concerning the order of rules application. It is therefore a parameter of the program and can be easily changed. After having experimented the prototype on some examples, it turned out that a good strategy was to apply the rules **Rewrite**$_1$, **Rewrite**$_2$ and **Induce** in any order provided that **Trivial** is applied after each step to constrain the search space as much as possible. The current version therefore adopts the following order, as shown by the code.

```
apply_all [trivial; rewrite1; trivial; rewrite2; trivial; induce] seq
```

Equality orientation. The current implementation uses the simple subterm ordering to orient equalities. It does however not apply to every equality, such as $x+y = y+x$ for instance. We made here the choice to arbitrarily orient such equalities from left to right.

Narrowing position. When applying the **Induce** rule, there may be several potential positions where to perform the narrowing step, as is the case in the example of section 2 for instance. We chose some heuristics here: the position where narrowing applies is the one that entails the biggest substitution.

Rewrite position. The last nondeterminism source of the algorithm is the choice of the position where **Rewrite**$_2$ applies. A first solution would be to rewrite the goal using as many rewrite rules of $\mathcal{RE}_2$ as possible at the same time. This is however not a good strategy since goals like $S(x + y) = S(y + x)$ would be swapped into $S(y + x) = S(x + y)$ by the rewrite rule $x + y \to y + x$ and draw IndNarrow into an infinite loop. A better solution is to only rewrite one redex at each application of **Rewrite**$_2$. One has then to choose between several positions. Choosing it in a predetermined way, say the leftmost-innermost one, would not be fair though. That is why we chose to introduce some entropy here by randomly picking the position where the rewrite step shall occur.

Despite of the simplicity of the chosen ordering, as well as the naive choices made to tackle nondeterminism, the prototype solves a pretty wide range of goals out of the box. Among them commutativity of addition and the fact that the mutually recursive and straightforward definitions of *even* coincide. It can be downloaded at `http://www.loria.fr/~brauner/rew.ml`.

7.3 Comparison with other methods

The following section contributes to illustrate the well-known fact that in implicit induction, the selection of good inductive positions can lead to dramatic improvements. Let us return to our motivating example (section 2). We can easily show the conjecture with our system (subsection 7.3.1). Informally speaking, the proof attempt succeeds, because the narrowing step is performed at the "good position". Conversely, it fails either if too many variables are replaced, or if the narrowing step is performed at any "bad" position. We show that it is respectively the case with Spike-AC [BER 96] (subsection 7.3.2), and with another method based on narrowing at *defined-innermost* positions [AOT 06] (subsection 7.3.3). We finally compare our work with the "descente infinie" approach (subsection 7.3.4).

7.3.1 An AC-example

Let us consider the specification **Simple arithmetic** (Fig 1) and $<$ be any rewrite path ordering showing the termination of its set of rules. Assume that $+$ and $*$ are AC-symbols, and the following goal:

$$\boxed{\emptyset, AC(<>)|\Gamma_2 \vdash_{\mathcal{RE}_1|\emptyset} exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

– Let us apply **InduceAC** at position 2.1.

l	$CSUC_{AC(+,*)}(exp(X,N), l)$
$exp(x, 0)$	$\sigma_1 = \{X \to X_1; N \to 0; x \to X_1\}$
$exp(x, s(y))$	$\sigma_2 = \{X \to X_1; N \to s(N_1);$ $x \to X_1; y \to N_1\}$

– After normalization, we obtain the subgoals:

$\emptyset, AC(+,*)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_1}$ $s(0) \approx s(0)$
$\emptyset, AC(+,*)\|\Gamma_2 \vdash_{\mathcal{RE}_1\|\mathcal{RE}_{ind}(Q)\sigma_2}$ $X_1 * Y * exp(X_1 * Y, N_1) \approx$ $X_1 * exp(X_1, N_1) * Y * exp(Y, N_1)$

– **TrivialAC** gets rid of this first subgoal, and since $N_1 < s(N_1)$, the induction hypothesis can be applied to the second one.

7.3.2 The example with a method based on inductive schemes (Spike-AC: [BER 96])

In this framework, induction schemes are defined first by a function that, given a conjecture, selects the positions of variables where induction will be applied, and second by a special set of terms called a *test set*. In this case, the induction variables are *all* the variables: X, Y and N. However, there is a test instance which cannot be simplified by the induction hypothesis. This is the situation we described in Section 2.

– Simplify:

$$\frac{< E \cup \{s \approx t\}, \mathcal{H} >}{< E \cup \{s' \approx t\}, \mathcal{H} >} \; s \to_{\mathcal{R} \cup \mathcal{H}} s'$$

– Delete:

$$\frac{< E \cup \{s \approx t\}, \mathcal{H} >}{< E, \mathcal{H} >}$$

– Expand:

$$\frac{< E \cup \{s \approx t\}, \mathcal{H} >}{< E \cup Expd_u(s,t), \mathcal{H} \cup \{s \to t\} >} \; u \in DI(s), \; s \succ t$$

Fig. 7 Aoto's proof search system

7.3.3 The example with another method based on narrowing at defined-innermost *positions ([AOT 06]])*

Aoto [AOT 06] reports that Koike and Toyama [KOI 00] extracted an abstract principle of implicit induction in terms of abstract reduction systems:

Proposition 6 Let $\mathcal{R}$ and $\mathcal{H}$ be rewrite systems, and $\preceq$ be a well founded quasi-ordering on a set A. Suppose:

1. $\to_{\mathcal{R} \cup \mathcal{H}} \; \subseteq \; \succ$;
2. $\to_{\mathcal{H}} \; \subseteq \; \to_{\mathcal{R}} \circ \overset{*}{\to}_{\mathcal{R} \cup \mathcal{H}} \circ \overset{*}{\leftarrow}_{\mathcal{R} \cup \mathcal{H}}$;

then $\overset{*}{\leftrightarrow}_{\mathcal{R}} \; = \; \overset{*}{\leftrightarrow}_{\mathcal{R} \cup \mathcal{H}}$.

Aoto designed a proof search system (Figure 7) based on the above result. It starts by putting conjectures into a set E and letting $\mathcal{H} = \emptyset$. Then the procedure rewrites $< E, \mathcal{H} >$ by applying one of the inference rules. If it eventually becomes of the form $< \emptyset, \mathcal{H}' >$ then the procedure return "success" (this means that the conjectures are inductive theorems of the underlying rewrite system $\mathcal{R}$). Observe that the rule **Expand** is quite similar to our rule **Induce**. However, the narrowing step must be performed in the member of the goal which is greater. Recall that the conjecture was:

$$\boxed{exp(X * Y, N) \approx exp(X, N) * exp(Y, N)}$$

The first member of the goal is greater w.r.t. our ordering $<$. Thus, the narrowing step can be performed only at position 1.1. We obtain:

$$\boxed{exp(X_1 * Y_1 + X_1, N_1) \approx exp(X_1, N_1) * exp(s(Y_1), N_1)}$$

which cannot be simplified by the induction hypothesis. Notice that Aoto has also provided more sophisticated inference rules, but the narrowing step must always be performed in the member of the goal which is greater.

7.3.4 Induction in sequent calculus: another approach

Our system can be considered as a customised version of the framework provided in [DEP 02] tailored for equational goals towards A and AC theories. The aim of basing

this framework on deduction modulo was to undertake a rigorous proof-theoretic investigation of the proof principles of mathematical induction. We think that such an analysis is of special interest since it has implications for the provision of a proof term. An other choice has been to extend the language of standard first order logic with an inductive definition schema. More precisely, [BRO 06] has provided an extension of Gentzen's LK sequent calculus to obtain canonical sequent calculus proof systems with an inductive definition schema. This method uses the principle of "descente infinie" à la Fermat.

One way of stating the "descente infinie" principle is that in order to prove that P is true for every x, it suffices to demonstrate that if P is not true for a particular number x, then there is an infinite strictly decreasing chain $\ldots < x_n < \ldots < x_1 < x$ wrt a well founded relation $<$, which is impossible.

Example 2 [BRO 06] Let the sets E and O of even and odd numbers be given by the following inductive definition:

$$0 \in E \qquad n \in E \Rightarrow s(n) \in O \qquad n \in O \Rightarrow s(n) \in E$$

Informally, the justification of the result by "descente infinie" is as follows. Let n be an integer. Assume $n = 0$, then we are done. The other case is $n = s(m)$. If $m = 0$ then we are done as $n = s(0)$ is an odd number, so we need only consider the case where $m = s(m')$ for some natural number m'. By repeating this argument infinitely often, we are left only with the case in which we have an infinite descending sequence $n > m > m' > m'' > \ldots$, which leads to a contradiction, since natural numbers are well-ordered. Thus every natural number must indeed be either even or odd.

Let us see how this proof is formalised in [BRO 06]. First, the induction rule associated to the predicate "Peano integer" N is defined by:

$$\frac{\Gamma, t = 0 \vdash \Delta \quad \Gamma, t = sx, Nx \vdash \Delta}{\Gamma, Nt \vdash \Delta}\ \text{Case N}$$

Now, with the notations above, this proof has the following form:

$$\dfrac{\dfrac{\dfrac{}{\vdash E0, O0}\ ER_1}{x_0 = 0 \vdash Ex_0, Ox_0}\ = L \quad \dfrac{\dfrac{\dfrac{\vdots}{Nx_1 \vdash Ex_1, Ox_1}\ (\text{Case N})}{Nx_1 \vdash Ox_1, Osx_1}\ OR_1}{Nx_1 \vdash Esx_1, Osx_1}\ ER_2}{x_0 = sx_1, Nx_1 \vdash Ex_0, Ox_0}\ = L}{Nx_0 \vdash Ex_0, Ox_0}\ \text{Case N}$$

Observe that we obtain a tree with exactly one infinite branch. Informally speaking, the infinite tree above is a proof in this system because the inductive predicate N is "unfolded infinitely often" along the only infinite branch in the tree.

8 Conclusion

We have extended the inductive proof search method based on narrowing to the case where theories contain non-orientable axioms. The main inference rule is based on a restricted application of narrowing at defined-innermost positions, strongly motivated by the until now unnoticed fact that a restricted notion of equational unification, based only on constructors, very significantly reduces the number of unifiers to be considered. This general approach is proved correct and refutationaly complete. We then applied it to the specific case of rewriting modulo AC or A axioms and show on two examples how the method safely restricts the proof search space. This provides a significant improvement on previous inductive proof search approaches.

An interesting side result of our approach is the introduction of a new kind of E-unifiers that we called constructor E-unifiers. In the case of associative and/or commutative theories E, they have the nice property to considerably reduce the number of unifiers to be considered in a standard complete set of unifiers that may be huge or even infinite in these theories. A natural and challenging question is to build a unification theory for these specific unifiers.

First motivated by the wish to provide a bridge between explicit and implicit induction, our approach achieves this goal through a specific instance of the sequent calculus modulo [DOW 01] that clarifies the respective roles and uses of the noetherian induction principle and of equational rewriting. Although heuristics for lemma speculation, generalisation and induction rule choice are always in need of improvement for inductive proof search, it was not the aim of our work. For instance, a suitable noetherian ordering is implicitly assumed throughout the paper, rather than discovered by the search strategy like in explicit induction methods. We expect however to have an automated construction of inductive proofs into the sequent calculus for insertion into proof assistants. Summing up, we can say that this work takes place in the following constructive process:1) building a proof of an inductive conjecture with our system; 2) associating to this proof another proof in sequent calculus modulo; 3) translating the proof in sequent calculus modulo into a proof term, whose type is the initial conjecture. Our system is therefore designed to collaborate with other proof assistants in a safe way, in order: 1) to find the most convenient noetherian ordering and required lemmas; 2) to check the proof. We hope therefore that this work can provide proof assistants a theoretical foundation based on deduction modulo. An implementation in lemuridæ which is based on superdeduction modulo [BRA 07] will be an interesting follow up.

Acknowledgments This paper has been initiated by the works done with Eric Deplagne and we had still many constructive and useful discussions with him these last years on that topics. Many thanks also to Adel Bouhoula, Sorin Stratulat and Michael Rusinowitch for clarifying discussions on induction by rewriting, and to the referees for their careful reading and valuable suggestions.

References

ALL 08. Lisa Allali, and Paul Brauner. "A Semantic Normalization Proof for Inductive Types". Research report, 2008. 25

AOT 06. Takahito Aoto. "Dealing with Non-orientable Equations in Rewriting Induction". In F. Pfenning, éditeur, *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, volume 4098, pages 242–256, Nara (Japan), apr 2006. Lecture Notes in Computer Science. 3, 27, 28

AUT 99. Serge Autexier, Dieter Hutter, Heiko Mantel, and Axel Schairer. "System description: Inka 5.0 – a logic voyager". In Harald Ganzinger, éditeur, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, volume 1632 de *Lecture Notes in Artificial Intelligence*, pages 207–211, Trento, Italy, july 1999. Springer. 2

BAA 98. Franz Baader, and Tobias Nipkow. "*Term Rewriting and all That*". Cambridge University Press, 1998. 5

BER 96. Narjès Berregeb, Adel Bouhoula, and Michaël Rusinowitch. "Automated verification by induction with associative-commutative operators.". In Rajeev Alur, and Thomas A. Henzinger, éditeurs, *CAV*, volume 1102 de *Lecture Notes in Computer Science*, pages 220–231. Springer, 1996. 3, 27

BER 97. Narjès Berregeb. "*Preuves par induction implicite : cas des théories associatives-commutatives et observationnelles*". Thèse de Doctorat d'Université, Université Henri Poincaré – Nancy 1, June 1997. 4

BER 04. Yves Bertot, and Pierre Casteran. "*Interactive Theorem Proving and Program Development*". Springer-Verlag, 2004. 2

BOU 92. Adel Bouhoula, E. Kounalis, and M. Rusinowitch. "Spike: An automatic theorem prover". In *Proceedings of the 1st International Conference on Logic Programming and Automated Reasoning, St. Petersburg (Russia)*, volume 624 de *Lecture Notes in Artificial Intelligence*, pages 460–462. Springer-Verlag, July 1992. 2, 9, 12

BOU 95. Adel Bouhoula, and Michaël Rusinowitch. Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14(2):189–235, 1995. 3

BOY 79. R. S. Boyer, and J. Strother Moore. "*A Computationnal Logic*". ACM monograph series. Academic Press, Inc, 1979. 4

BRA 07. Paul Brauner, Clément Houtmann, and Claude Kirchner. "Principles of superdeduction". In Luke Ong, éditeur, *LICS '07: Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*, pages 41–50, Wroclaw, Poland, jul 2007. IEEE Computer Society. 25, 30

BRO 06. James Brotherston. "*Sequent Calculus Proof Systems for Inductive Definitions*". PhD thesis, University of Edinburgh, November 2006. 29

BRO 07. James Brotherston, and Alex Simpson. "Complete sequent calculi for induction and infinite descent". In *submitted to Logic in Computer Science*, jan 2007. 2

BUN 99. Alan Bundy. "The automation of proof by mathematical induction". In Alan Robinson, and Andrei Voronkov, éditeurs, *Handbook of automated reasonning*. Elsevier Science Publishers B. V. (North-Holland), 1999. 2

COM 00. Hubert Comon, and Robert Nieuwenhuis. Induction=i-axiomatization+first-order consistency. *Inf. Comput.*, 159(1-2):151–186, 2000. 2

COM 01. Hubert Comon. "Inductionless induction". In A. Robinson, and A. Voronkov, éditeurs, *Handbook of Automated Reasoning*, volume I, chapter 14, pages 914–959. Elsevier Science, 2001. 2

DEP 02. Eric Deplagne. "*Système de preuve modulo récurrence*". Thèse de doctorat, Université Nancy 1, November 2002. 3, 8, 9, 28

DEP 03. Eric Deplagne, Claude Kirchner, Hélène Kirchner, and Quang-Huy Nguyen. "Proof search and proof check for equational and inductive theorems". In Franz Baader, éditeur, *Proceedings of CADE-19*, Miami, Florida, July 2003. Springer-Verlag. 3

DEP 04. Eric Deplagne, and Claude Kirchner. "Induction as deduction modulo". Rapport de recherche, LORIA, Nov 2004. 8

DOW 01. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic. *Mathematical Structures in Computer Science*, 11(1):21–45, 2001. 8, 30

DOW 03. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, Nov 2003. 3, 8

FRI 86. Laurent Fribourg. "A strong restriction of the inductive completion procedure". In *Proceedings 13th International Colloquium on Automata, Languages and Programming*, volume 226 de *Lecture Notes in Computer Science*, pages 105–115. Springer-Verlag, 1986. 2

GIR 89. Jean-Yves Girard, Yves Lafont, and Paul Taylor. "*Proofs and Types*", volume 7 de *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1989. 9

GOG 80. Joseph A. Goguen. "How to prove algebraic inductive hypotheses without induction, with applications to the correctness of data type implementation". In W. Bibel, and R. Kowalski, éditeurs, *Proceedings 5th International Conference on Automated Deduction, Les Arcs (France)*, volume 87 de *Lecture Notes in Computer Science*, pages 356–373. Springer-Verlag, 1980. 2

HOU 08. Clément Houtmann. "Axiom directed Focusing". 2008. 25

HUL 80. Jean-Marie Hullot. "Canonical forms and unification". In *Proceedings 5th International Conference on Automated Deduction, Les Arcs (France)*, pages 318–334, July 1980. 8

JOU 86. Jean-Pierre Jouannaud, and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, 15(4):1155–1194, 1986. 3, 6

KAP 95. Deepak Kapur, and Hantao Zhang. An overview of rewrite rule laboratory (RRL). *J. Computer and Mathematics with Applications*, 29(2):91–114, 1995. 2, 9, 12

KAU 96. Matt Kaufmann, and J Strother Moore. "ACL2: An industrial strength version of nqthm". In *Compass'96: Eleventh Annual Conference on Computer Assurance*, page 23, Gaithersburg, Maryland, 1996. National Institute of Standards and Technology. 2

KIR 99. Claude Kirchner, and Hélène Kirchner. Rewriting, solving, proving. A preliminary version of a book available at `www.loria.fr/~ckirchne/rsp.ps.gz`, 1999. 5, 8

KIR 06. Claude Kirchner, Hélène Kirchner, and Fabrice Nahon. "Narrowing based inductive proof search: Definition and optimisations". Research report, LORIA, Mars 2006. 3

KIR 07. Florent Kirchner. "A finite first-order theory of classes". In Thorsten Altenkirch, and Conor McBride, éditeurs, *Proc. 2006 Int. Workshop on Types for Proofs and Programs*, volume 4502 de *Lecture notes in Computer Science*, pages 188–202. Springer-Verlag, 2007. 25

KNU 70. Donald E. Knuth, and P. B. Bendix. "Simple word problems in universal algebras". In J. Leech, éditeur, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970. 2

KOI 00. H Koike, and Y. Toyama. Inductionless induction and rewriting induction. *Computer Software*, 17(6):1–12, 2000. 28

MAR 93. Claude Marché. "*Réécriture modulo une théorie présentée par un système convergent et décidabilité du problème du mot dans certaines classes de théories équationnelles*". Thèse de Doctorat d'Université, Université de Paris-Sud, Orsay (France), October 1993. 17, 18, 20, 21

MUS 80. David R. Musser. "On proving inductive properties of abstract data types". In *Proceedings 7th ACM Symp. on Principles of Programming Languages*, pages 154–162. ACM, 1980. 2

NAH 07. Fabrice Nahon. "*Preuve par induction dans le calcul des séquents modulo*". PhD thesis, Université Henri Poincaré - Nancy I, 2007. 7, 14, 15, 16, 17, 18, 20, 21

NIP 02. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. "*Isabelle/HOL — A Proof Assistant for Higher-Order Logic*", volume 2283 de *Lecture Notes in Computer Science*. Springer-Verlag, 2002. 2

PET 81. Gerald E. Peterson, and Mark E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, 28:233–264, 1981. 3

RED 90. Udday Reddy. "Term rewriting induction". In M. E. Stickel, éditeur, *Proceedings 10th International Conference on Automated Deduction, Kaiserslautern (Germany)*, volume 449 de *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 1990. 2, 3

URB 01. Christian Urban. "Strong normalisation for a Gentzen-like cut-elimination procedure". In *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, pages 415–430. Springer-Verlag, 2001. 25

VIS 01. Eelco Visser. "Stratego: A language for program transformation based on rewriting strategies. System description of Stratego 0.5". In A. Middeldorp, éditeur, *Rewriting Techniques and Applications (RTA'01)*, volume 2051 de *Lecture Notes in Computer Science*, pages 357–361. Springer-Verlag, May 2001. 26

WEC 92. Wolfgang Wechler. "*Universal Algebra for Computer Scientists*", volume 25 de *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1992. 9, 11